

LABEL OLPA OLPA 00176366CC USER=00000000; REMOVE ESPOL/DISK;COMPILE ESPOL/DISK TSPOL LIBRARY;TSPOL TSPOL /ESPOL
BURROUGHS B-5700 TSPOL COMPILER MARK XVI.0.00 FRIDAY, 12/31/76, 1107 PM.

ESPOL /DISK
=====

```
#####
%
%           B-5700 ESPOL COMPILER
%           MARK XVI.0.00
%           OCT  1, 1974
%
%#####
%
%  COMMENT: * TITLE: B5500/B5700 MARK XVI SYSTEM RELEASE          *
%            * FILE ID: SYMBOL/ESPOL TAPE ID: SYMBOL1/FILE000      *
%            * THIS MATERIAL IS PROPRIETARY TO BURROUGHS CORPORATION *
%            * AND IS NOT TO BE REPRODUCED, USED, OR DISCLOSED     *
%            * EXCEPT IN ACCORDANCE WITH PROGRAM LICENSE OR UPON  *
%            * WRITTEN AUTHORIZATION OF THE PATENT DIVISION OF      *
%            * BURROUGHS CORPORATION, DETROIT, MICHIGAN 48232      *
%            *
%            * COPYRIGHT (C) 1971, 1972, 1974                      *
%            * BURROUGHS CORPORATION                                *
%            * AA320206 AA393180 AA332366                          *;
%  COMMENT#####
%            ERROR MESSAGES
%#####
%
%  ERROR NUMBER  ROUTINE:ERROR MESSAGE
%  000           BLOCK: DECLARATION NOT FOLLOWED BY SEMICOLON.
%  001           BLOCK: IDENTIFIER DECLARED TWICE IN SAME BLOCK.
%  002           PROCEDUREDEC: SPECIFICATION PART CONTAINS
%                  IDENTIFIER NOT APPEARING IN
%                  FORMAL PARAMETER PART.
%  003           BLOCK: NON-IDENTIFIER APPEARS IN IDENTIFIER
%                  LIST OF DECLARATION.
%  004           PROCEDUREDEC: STREAM PROCEDURE DECLARATION
%                  PRECEDED BY ILLEGAL DECLARATOR.
%  005           PROCEDUREDEC: PROCEDURE DECLARATION PRECEDED
%                  BY ILLEGAL DECLARATOR.
%  006           PROCEDUREDEC: PROCEDURE IDENTIFIER USED BEFORE
%                  IN SAME BLOCK(NOT FORWARD).
%  007           PROCEDUREDEC: PROCEDURE IDENTIFIER NOT FOLLOWED
%                  BY ( OR SEMICOLON IN PROCEDURE
%                  DECLARATION.
%  008           PROCEDUREDEC: FORMAL PARAMETER LIST NOT FOLLOWED
%                  BY ).
%  009           PROCEDUREDEC: FORMAL PARAMETER PART NOT FOLLOWED
%                  BY SEMICOLON.
%  010           PROCEDUREDEC: VALUE PART CONTAINS IDENTIFIER
%                  WHICH DID NOT APPEAR IN FORMAL
%                  PARAPART.

00001000 T 0000
00001010 T 0000
00001020 T 0000
00001030 T 0000
00001040 T 0000
00001050 T 0000
00001060 T 0000
00001070 T 0000
00001072 T 0000
00001073 T 0000
00001074 T 0000
00001075 T 0000
00001076 T 0000
00001077 T 0000
00001078 T 0000
00001079 T 0000
00001080 T 0000
00001081 T 0000
00001082 T 0000
00001110 T 0000
00001120 T 0000
00001130 T 0000
00001140 T 0000
00002000 T 0000
00003000 T 0000
00004000 T 0000
00005000 T 0000
00006000 T 0000
00007000 T 0000
00008000 T 0000
00009000 T 0000
00010000 T 0000
00011000 T 0000
00012000 T 0000
00013000 T 0000
00014000 T 0000
00015000 T 0000
00016000 T 0000
00017000 T 0000
00018000 T 0000
00019000 T 0000
00020000 T 0000
00021000 T 0000
00022000 T 0000
00023000 T 0000
00024000 T 0000
00025000 T 0000
```

011	PROCEDUREDEC:	VALUE PART NOT ENDED BY SEMICOLON,	00026000	T	0000
012	PROCEDUREDEC:	MISSING OR ILLEGAL SPECIFICATION	00027000	T	0000
		PART,	00028000	T	0000
013	PROCEDUREDEC:	OWN USED IN ARRAY SPECIFICATION,	00029000	T	0000
014	PROCEDUREDEC:	SAVE USED IN ARRAY SPECIFICATION,	00030000	T	0000
015	BLOCK:	DECLARATION PRECEDED BY ILLEGAL DECLARATOR,	00031000	T	0000
016	ARRAYDEC:	ARRAY ID IN DECLARATION NOT FOLLOWED	00032000	T	0000
		BY [.	00033000	T	0000
017	ARRAYDEC:	LOWER BOUND IN ARRAY DEC NOT	00034000	T	0000
		FOLLOWED BY [.	00035000	T	0000
018	ARRAYDEC:	BOUND PAIR LIST NOT FOLLOWED BY J,	00036000	T	0000
019	ARRAYSPEC:	ILLEGAL LOWER BOUND DESIGNATOR IN	00037000	T	0000
		ARRAY SPECIFICATION,	00038000	T	0000
020	BLOCK:	OWN APPEARS IMMEDIATELY BEFORE	00039000	T	0000
		IDENTIFIER(NO TYPE),	00040000	T	0000
021	BLOCK:	SAVE APPEARS IMMEDIATELY BEFORE	00041000	T	0000
		IDENTIFIER(NO TYPE),	00042000	T	0000
022	BLOCK:	STREAM APPEARS IMMEDIATELY BEFORE	00043000	T	0000
		IDENTIFIER(THE WORD PROCEDURE LEFT	00044000	T	0000
		OUT),	00045000	T	0000
023	BLOCK:	DECLARATOR PRECEDED ILLEGALLY BY	00046000	T	0000
		ANOTHER DECLARATOR,	00047000	T	0000
024	PROCEDUREDEC:	LABEL CANNOT BE PASSED TO FUNCTION,	00048000	T	0000
025	BLUCK:	DECLARATOR OR SPECIFIER ILLEGALLY	00049000	T	0000
		PRECEDED BY OWN OR SAVE OR SOME	00050000	T	0000
		OTHER DECLARATOR,	00051000	T	0000
026	FILEDEC:	MISSING (IN FILE DEC,	00052000	T	0000
027	FILEDEC:	NO. OF BUFFERS IN FILE DEC MUST BE	00053000	T	0000
		AN UNSIGNED INTEGER,	00054000	T	0000
028	FILEDEC:	ILLEGAL BUFFER PART OR SAVE FACTOR	00055000	T	0000
		IN FILE DEC,	00056000	T	0000
029	FILEDEC:	MISSING) IN FILE DEC,	00057000	T	0000
030	PROCEDUREDEC:	PROCEDURE TYPE AT ACTUAL DECLARATION	00058000	T	0000
		TIME DIFFERENT THAN AT FORWARD DEC,	00059000	T	0000
031	LISTDEC:	MISSING (IN LISTDEC,	00060000	T	0000
032	FORMATDEC:	MISSING (IN FORMAT DEC,	00061000	T	0000
033	SWITCHDEC:	SWITCH DEC DOES NOT HAVE + OR	00062000	T	0000
		FORWARD AFTER IDENTIFIER,	00063000	T	0000
034	SWITCHFILEDEC:	MISSING + AFTER FILED,	00064000	T	0000
035	SWITCHFILEDEC:	NON FILE ID APPEARING IN DECLARATION	00065000	T	0000
		OF SWITCHFILE,	00066000	T	0000
036	SUPERFORMATDEC:	FORMAT ID NOT FOLLOWED BY + .	00067000	T	0000
037	SUPERFORMATDEC:	MISSING (AT START OF FORMATPHRASE .	00068000	T	0000
038	SUPERFORMATDEC:	FORMAT SEGMENT >1022 WORDS,	00069000	T	0000
040	SEGMENT:	SAVE CODE EXCEEDS 4080 WHICH KERNAL CAN H/L	00069100	T	0000
050	ANYWHERE:	OUT OF RANGE OF C RELATIVE ADDRESSING FOR CONSTANT	00069500	T	0000
051	BLOCK :	ILLEGAL F RELATIVE ADDRESS EXP IN DECLARATION	00069510	T	0000
052	BLOCK:	PROCEDURE WHOSE BODY NOT A BLOCK	00069520	T	0000
053	ARRAYDEC:	CANT FIND RIGHT BRACKET IN SAVE ARRAY DEC	00069530	T	0000
054	ARRAYDEC:	FILL PART OF SAVE ARRAY DEC LONGER THAN SIZE	00069540	T	0000
056	ARRAYDEC:	ILLEGAL DIMENSION INDICATOR IN ARRAY DEC	00069560	T	0000
057	SEGMENTSTART:	SAVE STORAGE NOT ALLOWED WITH INTRINSIC OPTION	00069570	T	0000
098	IOSTMT:	ILLEGAL SPECIFIER IN SCOPE STMT: MUST BE 215,	00069980	T	0000
099	INLINE:	EXTRA : IN STREAM HEAD,	00069990	T	0000
100	ANYWHERE:	UNDECLARED IDENTIFIER,	00070000	T	0000
101	CHECKER:	AN ATTEMPT HAS BEEN MADE TO ADDRESS AN	00071000	T	0000
		IDENTIFIER WHICH IS LOCAL TO ONE PRUCEDURE AND GLOBAL	00072000	T	0000

	TO ANOTHER, IF THE QUANTITY IS A PROCEDURE NAME OR	00073000 T	0000
	AN OWN VARIABLE THIS RESTRICTION IS RELAXED.	00074000 T	0000
102	AEXP: CONDITIONAL EXPRESSION IS NOT OF ARITHMETIC TYPE	00075000 T	0000
103	PRIMARY: PRIMARY MAY NOT BEGIN WITH A QUANTITY OF THIS	00076000 T	0000
	TYPE.	00077000 T	0000
104	ANYWHERE: MISSING RIGHT PARENTHESIS.	00078000 T	0000
105	ANYWHERE: MISSING LEFT PARENTHESIS.	00079000 T	0000
106	PRIMARY: PRIMARY MAY NOT START WITH DECLARATOR.	00080000 T	0000
107	BEXP: THE EXPRESSION IS NOT OF BOOLEAN TYPE.	00081000 T	0000
108	EXPRS: A RELATION MAY NOT HAVE CONDITIONAL EXPRESSIONS	00082000 T	0000
	AS THE ARITHMETIC EXPRESSIONS.	00083000 T	0000
109	BOOSEC, SIMPBOO, AND BOOCOMP: THE PRIMARY IS NOT BOOLEAN.	00084000 T	0000
110	BOUCOMP: A NON-BOOLEAN OPERATOR OCCURS IN A BOOLEAN	00085000 T	0000
	EXPRESSION.	00086000 T	0000
111	BOOPRIM: NO EXPRESSION (ARITHMETIC, BOOLEAN, OR DESIGNA-	00087000 T	0000
	TIONAL) MAY BEGIN WITH A QUANTITY OF THIS TYPE.	00088000 T	0000
112	BOOPRIM: NO EXPRESSION (ARITHMETIC, BOOLEAN, OR DESIGNA-	00089000 T	0000
	TIONAL) MAY BEGIN WITH A DECLARATOR.	00090000 T	0000
113	PARSE: EITHER THE SYNTAX OR THE RANGE OF THE LITERALS FOR	00091000 T	0000
	A CONCATENATE OPERATOR IS INCORRECT.	00092000 T	0000
114	DOTSYNTAX: EITHER THE SYNTAX OR THE RANGE OF THE LITERALS	00093000 T	0000
	FOR A PARTIAL WORD DESIGNATOR IS INCORRECT.	00094000 T	0000
115	DEXP: THE EXPRESSION IS NOT OF DESIGNATIONAL TYPE.	00095000 T	0000
116	IFCLAUSE: MISSING THEN.	00096000 T	0000
117	BANA: MISSING LEFT BRACKET.	00097000 T	0000
118	BANA: MISSING RIGHT BRACKET.	00098000 T	0000
119	COMPOUNDTAIL: MISSING SEMICOLON OR END.	00099000 T	0000
120	COMPOUNDTAIL: MISSING END.	00100000 T	0000
121	ACTUALPARAPART: AN INDEXED FILE MAY BE PASSED BY NAME	00101000 T	0000
	ONLY AND ONLY TO A STREAM PROCEDURE - THE STREAM	00102000 T	0000
	PROCEDURE MAY NOT DO A RELEASE ON THIS TYPE PARA-	00103000 T	0000
	METER.	00104000 T	0000
122	ACTUALPARAPART: STREAM PROCEDURE MAY NOT HAVE AN	00105000 T	0000
	EXPRESSION PASSED TO IT BY NAME.	00106000 T	0000
123	ACTUALPARAPART: THE ACTUAL AND FORMAL PARAMETERS DO NOT	00107000 T	0000
	AGREE AS TO TYPE.	00108000 T	0000
124	ACTUALPARAPART: ACTUAL AND FORMAL ARRAYS DO NOT HAVE SAME	00109000 T	0000
	NUMBER OF DIMENSIONS.	00110000 T	0000
125	ACTUALPARAPART: STREAM PROCEDURES MAY NOT BE PASSED AS A	00111000 T	0000
	PARAMETER TO A PROCEDURE.	00112000 T	0000
126	ACTUALPARAPART: NO ACTUAL PARAMETER MAY BEGIN WITH A	00113000 T	0000
	QUANTITY OF THIS TYPE.	00114000 T	0000
127	ACTUALPARAPART: THIS TYPE QUANTITY MAY NOT BE PASSED TO A	00115000 T	0000
	STREAM PROCEDURE.	00116000 T	0000
128	ACTUALPARAPART: EITHER ACTUAL AND FORMAL PARAMETERS DO	00117000 T	0000
	NOT AGREE AS TO NUMBER, OR EXTRA RIGHT PARENTHESIS.	00118000 T	0000
129	ACTUALPARAPART: ILLEGAL PARAMETER DELIMITER.	00119000 T	0000
130	RELSESTMT: NO FILE NAME.	00120000 T	0000
131	DOSTMT: MISSING UNTIL.	00121000 T	0000
132	WHILESTMT: MISSING DO.	00122000 T	0000
133	LABELR: MISSING COLON.	00123000 T	0000
134	LABELR: THE LABEL WAS NOT DECLARED IN THIS BLOCK.	00124000 T	0000
135	LABELR: THE LABEL HAS ALREADY OCCURED.	00125000 T	0000
136	FORMATPHRASE: IMPROPER FORMAT EDITING PHRASE.	00126000 T	0000
137	FORMATPHRASE: A FORMAT EDITING PHRASE DOES NOT HAVE AN	00127000 T	0000
	INTEGER WHERE AN INTEGER IS REQUIRED.	00128000 T	0000
138	FORMATPHRASE: THE WIDTH IS TOO SMALL IN E OR F EDITING	00129000 T	0000

	PHRASE,	00130000 T	0000
139	TABLE: DEFINE IS NESTED MORE THAN EIGHT DEEP,	00131000 T	0000
140	NEXTENT: AN INTEGER IN A FORMAT IS GREATER THAN 1023.	00132000 T	0000
141	SCANNER: INTEGER OR IDENTIFIER HAS MORE THAN 63	00133000 T	0000
	CHARACTORS,	00134000 T	0000
142	DEFINEGEN: A DEFINE CONTAINS MORE THAN 2047 CHARACTORS	00135000 T	0000
	(BLANK SUPPRESSED),	00136000 T	0000
143	COMPOUNDTAIL: EXTRA END,	00137000 T	0000
144	STMT: NO STATEMENT MAY START WITH THIS TYPE IDENTIFIER,	00138000 T	0000
145	STMT: NO STATEMENT MAY START WITH THIS TYPE QUANTITY,	00139000 T	0000
146	STMT: NO STATEMENT MAY START WITH A DECLARATOR - MAY BE	00140000 T	0000
	A MISSING END OF A PROCEDURE OR A MISPLACED	00141000 T	0000
	DECLARATION,	00142000 T	0000
147	SWITCHGEN: MORE THAN 256 EXPRESSIONS IN A SWITCH	00143000 T	0000
	DECLARATION,	00144000 T	0000
148	GETSPACE: MORE THAN 1023 PROGRAM REFERENCE TABLE CELLS	00145000 T	0000
	ARE REQUIRED FOR THIS PROGRAM,	00146000 T	0000
149	GETSPACE: MORE THAN 255 STACK CELLS ARE REQUIRED FOR THIS	00147000 T	0000
	PROCEDURE,	00148000 T	0000
150	ACTUALPARAPART: CONSTANTS MAY NOT BE PASSED BY NAME TO	00149000 T	0000
	STREAM PROCEDURES,	00150000 T	0000
151	FORSTMT: IMPROPER FOR INDEX VARIABLE,	00151000 T	0000
152	FORSTMT: MISSING LEFT ARROW FOLLOWING INDEX VARIABLE,	00152000 T	0000
153	FORSTMT: MISSING UNTIL OR WHILE IN STEP ELEMENT,	00153000 T	0000
154	FORSTMT: MISSING DO IN FOR CLAUSE,	00154000 T	0000
155	IFEXPI: MISSING ELSE	00155000 T	0000
156	LISTELEMENT: A DESIGNATIONAL EXPRESSION MAY NOT BE A LIST	00156000 T	0000
	ELEMENT,	00157000 T	0000
157	LISTELEMENT: A ROW DESIGNATOR MAY NOT BE A LISTELEMENT	00158000 T	0000
158	LISTELEMENT: MISSING RIGHT BRACKET IN GROUP OF ELEMENTS	00159000 T	0000
159	PROCSTMT: ILLEGAL USE OF PROCEDURE OR FUNCTION IDENTIFIER	00160000 T	0000
160	PURGE: DECLARED LABEL DOES NOT OCCUR,	00161000 T	0000
161	PURGE: DECLARED FORWARD PROCEDURE DOES NOT OCCUR,	00162000 T	0000
163	ZIPSTMT: MISSING COMMA IN ZIP STATEMENT	00163000 T	0000
163	FORMATPHRASE: THE WIDTH OF A FIELD IS MORE THAN 63,	00164000 T	0000
200	EMIT: SEGMENT TOO LARGE (> 4093SYLLABLES),	00165000 T	0000
201	SIMPLE VARIABLE: PARTIAL WORD DESIGNATOR NOT LEFT=MOST	00166000 T	0000
	IN A LEFT PART LIST,	00167000 T	0000
202	SIMPLE VARIABLE: MISSING , OR + ,	00168000 T	0000
203	SUBSCRIPTED VARIABLE: WRONG NUMBER OF SUBSCRIPTS IN A ROW	00169000 T	0000
	DESIGNATOR,	00170000 T	0000
204	SUBSCRIPTED VARIABLE: MISSING) IN A ROW DESIGNATOR,	00171000 T	0000
205	SUBSCRIPTED VARIABLE: A ROW DESIGNATOR APPEARS OUTSIDE OF	00172000 T	0000
	AN ACTUAL PARAMETER LIST OR FILL STATEMENT,	00173000 T	0000
206	SUBSCRIPTED VARIABLE: MISSING],	00174000 T	0000
207	SUBSCRIPTED VARIABLE: MISSING [,	00175000 T	0000
208	SUBSCRIPTED VARIABLE: WRONG NUMBER OF SUBSCRIPTS,	00176000 T	0000
209	SUBSCRIPTED VARIABLE: PARTIAL WORD DESIGNATOR NOT LEFT=	00177000 T	0000
	MOST IN A LEFT PART LIST,	00178000 T	0000
210	SUBSCRIPTED VARIABLE: MISSING , OR + ,	00179000 T	0000
211	VARIABLE: PROCEDURE ID USED OUTSIDE OF SCOPE IN LEFT PART,	00180000 T	0000
250	STREAM STMT: ILLEGAL STREAM STATEMENT,	00181000 T	0000
251	ANY STREAM STMT PROCEDURE: MISSING +,	00182000 T	0000
252	INDEX: MISSING + OR = ,	00183000 T	0000
253	INDEX: MISSING NUMBER OR STREAM VARIABLE,	00184000 T	0000
254	EMITC: NUMBER>63 OR NUMBER OF LABELS+LOCALS+FORMALS>63,	00185000 T	0000
255	DSS: MISSING STRING IN DS+ LIT STATEMENT,	00186000 T	0000

256	RELEASES: MISSING PARENTHESIS OR FILE IDENTIFIER IS NOT	00187000	T	0000
	A FORMAL PARAMETER.	00188000	T	0000
257	GOTOS, LABELS, OR JUMPS: LABEL SPECIFIED IS NOT ON THE SAME	00189000	T	0000
	NEST LEVEL AS A PRECEDING APPEARANCE OF THE	00190000	T	0000
	LABEL.	00191000	T	0000
258	LABELS: MISSING !.	00192000	T	0000
259	LABELS: LABEL APPEARS MORE THAN ONCE.	00193000	T	0000
260	GOTOS: MISSING LABEL IN A GO TO OR JUMP OUT TO STATEMENT.	00194000	T	0000
261	JUMPS: MISSING OUT IN JUMP OUT STATEMENT.	00195000	T	0000
262	NESTS: MISSING PARENTHESIS.	00196000	T	0000
263	IFS: MISSING SC IN IF STATEMENT.	00197000	T	0000
264	IFS: MISSING RELATIONAL IN IF STATEMENT.	00198000	T	0000
265	IFS: MISSING ALPHA, DC OR STRING IN IF STATEMENT.	00199000	T	0000
266	IFS: MISSING THEN IN IF STATEMENT.	00200000	T	0000
267	FREDFIX: THERE ARE GO TO STATEMENTS IN WHICH THE LABEL IS	00201000	T	0000
	UNDEFINED.	00202000	T	0000
268	EMITC: A REPEAT INDEX ≥ 64 WAS SPECIFIED OR TOO MANY	00203000	T	0000
	FORMAL PARAMETERS, LOCALS AND LABELS.	00204000	T	0000
269	TABLE: A CONSTANT IS SPECIFIED WHICH IS TOO LARGE	00205000	T	0000
	OR TOO SMALL.	00206000	T	0000
281	DBLSTMT: MISSING (.	00207000	T	0000
282	DBLSTMT: TOO MANY OPERATORS.	00208000	T	0000
283	DBLSTMT: TOO MANY OPERANDS.	00209000	T	0000
284	DBLSTMT: MISSING , .	00210000	T	0000
285	DBLSTMT: MISSING) .	00211000	T	0000
300	FILLSTMT: THE IDENTIFIER FOLLOWING "FILL" IS NOT	00212000	T	0000
	AN ARRAY IDENTIFIER.	00213000	T	0000
301	FILLSTMT: MISSING "WITH" IN FILL STATEMENT.	00214000	T	0000
302	FILLSTMT: IMPROPER FILL ELEMENT.	00215000	T	0000
303	FILLSTMT: NON-OCTAL CHARACTER IN OCTAL FILL.	00216000	T	0000
304	FILLSTMT: IMPROPER ARRAY ROW DESIGNATOR IN FILL.	00217000	T	0000
305	FILLSTMT: DATA IN FILL EXCEEDS 1023 WORDS.	00218000	T	0000
306	FILLSTMT: ODD NUMBER OF PARENTHESES IN FILL.	00218110	T	0000
400	MERRIMAC: MISSING FILE ID IN MONITOR DEC.	00219000	T	0000
401	MERRIMAC: MISSING LEFT PARENTHESIS IN MONITOR DEC.	00220000	T	0000
402	MERRIMAC: IMPROPER SUBSCRIPT FOR MONITOR LIST ELEMENT.	00221000	T	0000
403	MERRIMAC: IMPROPER SUBSCRIPT EXPRESSION DELIMITER IN	00222000	T	0000
	MONITOR LIST ELEMENT.	00223000	T	0000
404	MERRIMAC: IMPROPER NUMBER OF SUBSCRIPTS IN MONITOR LIST	00224000	T	0000
	ELEMENT.	00225000	T	0000
405	MERRIMAC: LABEL OR SWITCH MONITORED AT IMPROPER LEVEL.	00226000	T	0000
406	MERRIMAC: IMPROPER MONITOR LIST ELEMENT.	00227000	T	0000
407	MERRIMAC: MISSING RIGHT PARENTHESIS IN MONITOR DECLARATION.	00228000	T	0000
408	MERRIMAC: IMPROPER MONITOR DECLARATION DELIMITER.	00229000	T	0000
409	DMUP: MISSING FILE IDENTIFIER IN DUMP DECLARATION.	00230000	T	0000
410	DMUP: MISSING LEFT PARENTHESIS IN DUMP DECLARATION.	00231000	T	0000
411	DMUP: SUBSCRIPTED VARIABLE IN DUMP LIST HAS WRONG NUMBER OF	00232000	T	0000
	SUBSCRIPTS.	00233000	T	0000
412	DMUP: SUBSCRIPTED VARIABLE IN DUMP LIST HAS WRONG NUMBER OF	00234000	T	0000
	SUBSCRIPTS.	00235000	T	0000
413	DMUP: IMPROPER ARRAY DUMP LIST ELEMENT.	00236000	T	0000
414	DMUP: ILLEGAL DUMP LIST ELEMENT.	00237000	T	0000
415	DMUP: MORE THAN 100 LABELS APPEAR AS DUMP LIST ELEMENTS	00238000	T	0000
	IN ONE DUMP DECLARATION.	00239000	T	0000
416	DMUP: ILLEGAL DUMP LIST ELEMENT DELIMITER.	00240000	T	0000
417	DMUP: MISSING DUMP LABEL IN DUMP DECLARATION.	00241000	T	0000
418	DMUP: MISSING COLON IN DUMP DECLARATION.	00242000	T	0000

419	DMUPTIMPROPER DUMP DECLARATION DELIMITER.	00243000	T	0000
420	READSTMTIMISSING LEFT PARENTHESIS IN READ STATEMENT.	00244000	T	0000
421	READSTMTIMISSING LEFT PARENTHESIS IN READ REVERSE STATEMENT.	00245000	T	0000
422	READSTMTIMISSING FILE IN READ STATEMENT.	00246000	T	0000
423	READSTMTIMPROPER RELEASE INDICATOR.	00247000	T	0000
424	READSTMTIMPROPER FILE DELIMITER IN READ STATEMENT	00248000	T	0000
425	READSTMTIMPROPER FORMAT DELIMITER IN READ STATEMENT.	00249000	T	0000
426	READSTMTIMPROPER DELIMITER FOR SECOND PARAMETER IN READ STATEMENT.	00250000	T	0000
427	READSTMTIMPROPER ROW DESIGNATOR IN READ STATEMENT.	00251000	T	0000
428	READSTMTIMPROPER ROW DESIGNATOR DELIMITER IN READ STATEMENT.	00252000	T	0000
429	READSTMTIMISSING ROW DESIGNATOR IN READ STATEMENT.	00253000	T	0000
430	READSTMTIMPROPER DELIMITER PRECEEDING THE LIST IN A READ STATEMENT.	00254000	T	0000
431	HANDLETHETAILENDOFAREADORSPEACESTATEMENTIMPROPER END OF FILE LABEL IN READ OR SPACE STATEMENT.	00255000	T	0000
432	HANDLETHETAILENDOFAREADORSPEACESTATEMENTIMPROPER PARITY LABEL IN READ OR SPACE STATEMENT.	00256000	T	0000
433	HANDLETHETAILENDOFAREADORSPEACESTATEMENTIMISSING RIGHT BRACKET IN READ OR SPACE STATEMENT.	00257000	T	0000
434	SPACESTMTIMISSING LEFT PARENTHESIS IN SPACE STATEMENT.	00258000	T	0000
435	SPACESTMTIMPROPER FILE IDENTIFIER IN SPACE STATEMENT.	00259000	T	0000
436	SPACESTMTIMISSING COMMA IN SPACE STATEMENT.	00260000	T	0000
437	SPACESTMTIMISSING RIGHT PARENTHESIS IN SPACE STATEMENT.	00261000	T	0000
438	WRITESTMTIMISSING LEFT PARENTHESIS IN A WRITE STATEMENT.	00262000	T	0000
439	WRITESTMTIMPROPER FILE IDENTIFIER IN A WRITE STATEMENT.	00263000	T	0000
440	WRITESTMTIMPROPER DELIMITER FOR FIRST PARAMETER IN A WRITE STATEMENT.	00264000	T	0000
441	WRITESTMTIMISSING RIGHT BRACKET IN CARRIAGE CONTROL PART OF A WRITE STATEMENT.	00265000	T	0000
442	WRITESTMTIILLEGAL CARRIAGE CONTROL DELIMITER IN A WRITE STATEMENT.	00266000	T	0000
443	WRITESTMTIMPROPER SECOND PARAMETER DELIMITER IN WRITE STATEMENT.	00267000	T	0000
444	WRITESTMTIMPROPER ROW DESIGNATOR IN A WRITE STATEMENT.	00268000	T	0000
445	WRITESTMTIMISSING RIGHT PARENTHESIS AFTER A ROW DESIGNATOR IN A WRITE STATEMENT.	00269000	T	0000
446	WRITESTMTIMISSING ROW DESIGNATOR IN A WRITE STATEMENT.	00270000	T	0000
447	WRITESTMTIMPROPER DELIMITER PRECEEDING A LIST IN A WRITE STATEMENT.	00271000	T	0000
448	WRITESTMTIMPROPER LIST DELIMITER IN A WRITE STATEMENT.	00272000	T	0000
449	READSTMTIMPROPER LIST DELIMITER IN A READ STATEMENT.	00273000	T	0000
450	LOCKSTMTIMISSING LEFT PARENTHESIS IN A LOCK STATEMENT.	00274000	T	0000
451	LOCKSTMTIMPROPER FILE PART IN A LOCK STATEMENT.	00275000	T	0000
452	LOCKSTMTIMISSING COMMA IN A LOCK STATEMENT.	00276000	T	0000
453	LOCKSTMTIMPROPER UNIT DISPOSITION PART IN A LOCK STATEMENT.	00277000	T	0000
454	LOCKSTMTIMISSING RIGHT PARENTHESIS IN A LOCK STATEMENT.	00278000	T	0000
455	CLOSESTMTIMISSING LEFT PARENTHESIS IN A CLOSE STATEMENT.	00279000	T	0000
456	CLOSESTMTIMPROPER FILE PART IN A CLOSE STATEMENT.	00280000	T	0000
457	CLOSESTMTIMISSING COMMA IN A CLOSE STATEMENT.	00281000	T	0000
458	CLOSESTMTIMPROPER UNIT DISPOSITION PART IN A CLOSE STATEMENT.	00282000	T	0000
459	CLOSESTMTIMISSING RIGHT PARENTHESIS IN A CLOSE STATEMENT.	00283000	T	0000
460	RWNDSTMTIMISSING LEFT PARENTHESIS IN A REWIND STATEMENT.	00284000	T	0000
		00285000	T	0000
		00286000	T	0000
		00287000	T	0000
		00288000	T	0000
		00289000	T	0000
		00290000	T	0000
		00291000	T	0000
		00292000	T	0000
		00293000	T	0000
		00294000	T	0000
		00295000	T	0000
		00296000	T	0000
		00297000	T	0000
		00298000	T	0000
		00299000	T	0000

461	RWNDSTMT:IMPROPER FILE PART IN A REWIND STATEMENT.	00300000	T	0000
462	RWNDSTMT:MISSING RIGHT PARENTHESIS IN A REWIND STATEMENT.	00301000	T	0000
463	BLOCK:A MONITOR DECLARATION APPEARS IN THE SPECIFICATION PART OF A PROCEDURE.	00302000	T	0000
464	BLOCK:A DUMP DECLARATION APPEARS IN THE SPECIFICATION PART OF A PROCEDURE.	00303000	T	0000
465	INLINE: MISSING PARAMETER IDENTIFIER IN INLINE STREAM STATEMENT PARAMETER LIST.	00304000	T	0000
500	.ID: NEEDS DOUBLE PERIOD FOR PRTE IF PAST 512	00305000	T	0000
520	TABLE: STRING LONGER THAN ONE WORD (48 BITS).	00305001	T	0000
521	TABLE: STRING CONTAINS A NON-PERMISSIBLE CHARACTER.	00305002	T	0000
600	DOLLARCARD: NUMBER EXPECTED.	00305100	T	0000
601	DOLLARCARD: OPTION IDENTIFIER EXPECTED.	00306200	T	0000
602	DOLLARCARD: TOO MANY USER-DEFINED OPTIONS.	00306300	T	0000
603	DOLLARCARD: UNRECOGNIZED WORD OR CHARACTER.	00400000	T	0000
604	DOLLARCARD: MISMATCHED PARENTHESES.	00401000	T	0000
605	DOLLARCARD: \$ IN CARD COLUMN 1 FOR OMIT CARD	00403000	T	0000
610	READACARD: SEQUENCE ERROR.	00404000	T	0000
611	READACARD: ERROR LIMIT HAS BEEN EXCEEDED.	00405000	T	0000
	; BEGIN COMMENT OUTERMOST BLOCK;	00406000	T	0000
	PRT(22) = *LIST, LABEL, OR SEGMENT DESCRIPTOR*	00410000	T	0000
	PRT(23) = *OUTER BLOCK DESCRIPTOR*	00411000	T	0000
	PRT(24) = *SEGMENT DESCRIPTOR*	00490000	T	0000
		00500000	T	0000
	INTEGER ERRORCOUNT; COMMENT NUMBER OF ERROR MSGS. MCP WILL TYPE			
PRT(25) = ERRORCOUNT		START OF SEGMENT *****		2
	INTEGER SAVETIME; SYNTAX ERR AT EOJ IF THIS IS NON-ZERO, MUST BE @R+25;	00501000	T	0000
PRT(26) = SAVETIME	COMMENT SAVE-FACTOR FOR CODE FILE, GIVEN BY MCP.	00502000	T	0000
	IF COMPILE & GO =0, FOR SYNTAX, =-1, MUST BE AT R+26;	00503000	T	0000
PRT(27) = CARDNUMBER	INTEGER CARDNUMBER; % SEQ # OF CARD BEING PROCESSED.	00504000	T	0000
	INTEGER CARDCOUNT; % NUMBER OF CARDS PROCESSED.	00504100	T	0000
PRT(30) = CARDCOUNT	BOOLEAN BUILDLINE;	00504150	T	0000
PRT(31) = BUILDLINE	COMMENT RR1-RR11 ARE USED IN SOME PROCEDURES IN PLACE OF LOCALS TO SAVE STACK SPACE;	00504700	T	0000
	REAL RR1,RR2,RR3,RR4,RR5,RR6,RR7,RR8,RR9,RR10,RR11;	00505000	T	0000
PRT(32) = RR1		00506000	T	0000
PRT(33) = RR2		00507000	T	0000
PRT(34) = RR3				
PRT(35) = RR4				
PRT(36) = RR5				
PRT(37) = RR6				
PRT(40) = RR7				
PRT(41) = RR8				
PRT(42) = RR9				
PRT(43) = RR10				
PRT(44) = RR11				
	COMMENT SOME OF THE RRI ARE USED TO PASS FILE INFORMATION TO THE MAIN BLOCK;	00508000	T	0000
	COMMENT EXAMIN RETURNS THE CHARACTER AT ABSOLUTE ADDRESS NCR;	00509000	T	0000
	REAL STREAM PROCEDURE EXAMIN(NCR); VALUE NCR;	00510000	T	0000
PRT(45) = EXAMIN	BEGIN SI+NCR;DI+LOC EXAMIN;DI+DI+7; DS+CHR END;	00511000	T	0000
		00512000	T	0000

```

PRT(46) = GETF
      INTEGER STREAM PROCEDURE GETF(Q) VALUE Q;
      BEGIN SI+LOC GETF; SI+SI=7;DI+LOC Q;DI+DI+5;
        SKIP 3 DB; 9(IF SB THEN DS+SET ELSE DS+RESET; SKIP SB);
        DI+LOC Q;SI+Q;DS+WDS;SI+Q;GETF+SI
      END GETF;

```

```

00523000 T 0002
00524000 T 0002
00525000 T 0003
00526000 T 0005
00527000 T 0006

```

```

      COMMENT START SETTING UP FILE PARAMETERS;
      IF EXAMIN(RR11+GETF(3)+"Y08") #12 THEN RR1+5 ELSE
      BEGIN RR1+2;RR2+150 END;
      IF EXAMIN(RR11+5) #12 THEN RR3+4 ELSE
      BEGIN RR3+2; RR4+150 END;
      IF EXAMIN(RR11+10)=12 THEN
      BEGIN RR5+2;RR6+10;RR7+150 END ELSE
      BEGIN RR5+1;RR6+56;RR7+10 END;
      IF EXAMIN(RR11+15)=12 THEN
      BEGIN RR8+10;RR9+150 END ELSE
      BEGIN RR8+56;RR9+10 END;

```

```

00528000 T 0007
00529000 T 0007
00530000 T 0013
00531000 T 0016
00532000 T 0020
00533000 T 0022
00534000 T 0024
00535000 T 0027
00536000 T 0029
00537000 T 0032
00538000 T 0034
01000000 T 0036
01000800 T 0036

```

```

      BEGIN COMMENT MAIN BLOCK;
      INTEGER OPINX; % USED FOR INDEXING INTO OPTIONS ARRAY.
PRT(47) = *SEGMENT DESCRIPTOR*

```

START OF SEGMENT ***** 3

```

PRT(50) = OPINX
      BOOLEAN SETTING; % USED BY DOLLARCARD FOR AN OPTION'S SETTING.
PRT(51) = SETTING
      INTEGER NEWINX, ADDVALUE, BASENUM, TOTALNO;

```

```

01000802 T 0000
01000860 T 0000

```

```

PRT(52) = NEWINX
PRT(53) = ADDVALUE
PRT(54) = BASENUM
PRT(55) = TOTALNO

```

```

01000902 T 0000
01000904 T 0000

```

```

      DEFINE OPARSIZE = 200 #;
      ARRAY OPTIONS[0:OPARSIZE];
PRT(56) = OPTIONS

```

```

01000910 T 0002

```

```

      BOOLEAN OPTIONWORD;

```

```

PRT(57) = OPTIONWORD

```

```

      DEFINE
        CHECKBIT      = 1#,
        DEBUGBIT      = 2#,
        DECKBIT       = 3#,
        FORMATBIT     = 4#,
        INTBIT        = 5#,
        LISTABIT       = 6#,
        LISTBIT        = 7#,
        LISTPBIT       = 8#,
        MCPBIT         = 9#,
        MERGEBIT       = 10#,
        NESTBIT        = 11#,
        NEWBIT         = 12#,
        NEWINCLBIT     = 13#,
        OMITBIT        = 14#,
        PRINTDOLLARBIT = 15#,

```

```

01000920 T 0002
01000930 T 0002
01000940 T 0002
01000950 T 0002
01000960 T 0002
01000970 T 0002
01000980 T 0002
01000990 T 0002
01001000 T 0002
01001010 T 0002
01001020 T 0002
01001030 T 0002
01001040 T 0002
01001050 T 0002
01001060 T 0002

```

	PRTBIT	= 16#;	01001070	T	0002
	PUNCHBIT	= 17#;	01001080	T	0002
	PURGEBIT	= 18#;	01001090	T	0002
	SEGSBIT	= 19#;	01001100	T	0002
	SEQBIT	= 20#;	01001110	T	0002
	SEQERRBIT	= 21#;	01001120	T	0002
	SINGLBIT	= 22#;	01001130	T	0002
	STUFFBIT	= 23#;	01001140	T	0002
	VOIDBIT	= 24#;	01001150	T	0002
	VOIDTBIT	= 25#;	01001160	T	0002
	USEROPINX	= 26#;	01001170	T	0002
COMMENT	IF A NEW COMPILER-DEFINED OPTION IS ADDED, CHANGE USEROPINX		01001180	T	0002
	AND ADD OPTION IN DEFINES BELOW, IN DOLLARCARD, AND IN		01001190	T	0002
	FILL STATEMENT IN INITIALIZATION OF COMPILER;		01001200	T	0002
DEFINE	CHECKTOG	= OPTIONWORD.[CHECKBIT:1] #,	01001210	T	0002
	DEBUGTOG	= OPTIONWORD.[DEBUGBIT:1] #,	01001220	T	0002
	DECKTOG	= OPTIONWORD.[DECKBIT:1] #,	01001230	T	0002
	FORMATOG	= OPTIONWORD.[FORMATBIT:1] #,	01001240	T	0002
	INTOG	= OPTIONWORD.[INTBIT:1] #,	01001250	T	0002
	LISTATOG	= OPTIONWORD.[LISTABIT:1] #,	01001260	T	0002
	LISTOG	= OPTIONWORD.[LISTBIT:1] #,	01001270	T	0002
	LISTPTOG	= OPTIONWORD.[LISTPBIT:1] #,	01001280	T	0002
	MCPTOG	= OPTIONWORD.[MCPBIT:1] #,	01001290	T	0002
	MERGETOG	= OPTIONWORD.[MERGEBIT:1] #,	01001300	T	0002
	NESTOG	= OPTIONWORD.[NESTBIT:1] #,	01001310	T	0002
	NEWTOG	= OPTIONWORD.[NEWBIT:1] #,	01001320	T	0002
	NEWINCL	= OPTIONWORD.[NEWINCLBIT:1] #,	01001330	T	0002
	OMITTING	= OPTIONWORD.[OMITBIT:1] #,	01001340	T	0002
	PRINTDOLLARTOG	= OPTIONWORD.[PRINTDOLLARBIT:1] #,	01001350	T	0002
	PRTOG	= OPTIONWORD.[PRTBIT:1] #,	01001360	T	0002
	PUNHTOG	= OPTIONWORD.[PUNCHBIT:1] #,	01001370	T	0002
	PURGETOG	= OPTIONWORD.[PURGEBIT:1] #,	01001380	T	0002
	SEGSTOG	= OPTIONWORD.[SEGSBIT:1] #,	01001390	T	0002
	SEQTOG	= OPTIONWORD.[SEQBIT:1] #,	01001400	T	0002
COMMENT	SEQTOG INDICATES	RESEQUENCING IS TO BE DONE;	01001410	T	0002
	SEQERRTOG	= OPTIONWORD.[SEQERRBIT:1] #,	01001420	T	0002
	SINGLTOG	= OPTIONWORD.[SINGLBIT:1] #,	01001430	T	0002
	STUFFTOG	= OPTIONWORD.[STUFFBIT:1] #,	01001440	T	0002
	VOIDING	= OPTIONWORD.[VOIDBIT:1] #,	01001450	T	0002
	VOIDTAPE	= OPTIONWORD.[VOIDTBIT:1] #,	01001460	T	0002
	DUMMY	= #;	01001470	T	0002
	BOOLEAN NOHEADING;	% TRUE IF DATIME HAS NOT BEEN CALLED.	01001480	T	0002
PRT(60) = NOHEADING	BOOLEAN NEWBASE;	% NEW BASENUM FOUND ON A NEW S-CARD.	01001490	T	0002
PRT(61) = NEWBASE	BOOLEAN LASTCRDPATCH;	% NORMALLY FALSE, SET TO TRUE WHEN THE	01001500	T	0002
PRT(62) = LASTCRDPATCH		% LAST CARD FROM SYMBOLIC LIBRARY READ	01001510	T	0002
		% IS PATCHED FROM THE CARD READER.	01001520	T	0002
	INTEGER XMODE;	% TELLS DOLLARCARD HOW TO SET OPTIONS.	01001530	T	0002
PRT(63) = XMODE	BOOLEAN DOLLARTOG;	% TRUE IF SCANNING A DOLLAR CARD.	01001540	T	0002
PRT(64) = DOLLARTOG	INTEGER ERRMAX;	% COMPILATION STOPS IF EXCEEDED.	01001550	T	0002
PRT(65) = ERRMAX	BOOLEAN SEQXEQTOG;	% GIVE SEQ. NO. WHEN DS=ING OBJ.	01001560	T	0002
PRT(66) = SEQXEQTOG					

PRT(67) = LISTER	BOOLEAN LISTER; % LISTOG OR LISTATOG OR DEBUGTOG.	01001570 T	0002
PRT(70) = MEDIUM	ALPHA MEDIUM; % INPUT IS: T,C,P,CA,CB,CC.	01001580 T	0002
PRT(71) = MYCLASS	INTEGER MYCLASS; % USED IN DOLLARCARD EVALUATION.	01001590 T	0002
PRT(72) = BATMAN	REAL BATMAN; % USED IN DOLLARCARD EVALUATION.	01001600 T	0002
PRT(73) = SPECIAL	ARRAY SPECIAL[0:31];	01003000 T	0002
	COMMENT THIS ARRAY HOLDS THE INTERNAL CODE FOR THE SPECIAL	01004000 T	0005
	CHARACTORS; IT IS FILLED DURING INITIALIZATION;	01005000 T	0005
		01006000 T	0005
		01007000 T	0005
PRT(74) = INFO	ARRAY INFO [0:127,0:255];		
	COMMENT INFO CONTAINS ALL THE INFORMATION ABOUT A GIVEN IDENTIFIER	01008000 T	0007
	OR RESERVED WORD. THE FIRST WORD OF A GIVEN ENTRY IS	01009000 T	0007
	THE INTERNAL CODE (OR ELBAT WORD AS IT IS USUALLY	01010000 T	0007
	CALLED). THE SECOND WORD CONTAINS THE FORWARD BIT (IN	01011000 T	0007
	[1:1]) FOR PROCEDURES, THE LINK TO PREVIOUS ENTRY (IN	01012000 T	0007
	[4:8]), THE NUMBER OF CHARACTORS IN THE ALPHA REPRESENTA-	01013000 T	0007
	TION (IN [12:16]), AND THE FIRST 5 CHARACTORS OF ALPHA.	01014000 T	0007
	SUCCESSING WORDS CONTAIN THE REMAINING CHARACTORS OF ALPHA,	01015000 T	0007
	FOLLOWED BY ANY ADDITIONAL INFORMATION. THE ELBAT WORD	01016000 T	0007
	AND THE ALPHA FOR ANY QUANTITY ARE NOT SPLIT ACROSS A ROW	01017000 T	0007
	OF INFO. FOR PURPOSES OF FINDING AN IDENTIFIER OR	01018000 T	0007
	RESERVED WORD THE QUANTITIES ARE SCATTERED INTO 125	01019000 T	0007
	DIFFERENT LISTS OR STACKS, WHICH STACK CONTAINS A QUANTITY	01020000 T	0007
	IS GIVEN BY TAKING NAAAAA MOD 125 WHERE N IS THE NUMBER	01021000 T	0007
	OF CHARACTORS AND AAAAAA IS THE FIRST 5 CHARACTORS OF	01022000 T	0007
	ALPHA, FILLED IN WITH ZEROS FROM THE RIGHT IF NEEDED.	01023000 T	0007
	THIS NUMBER IS CALLED THE SCRAMBLE NUMBER OR INDEX.	01024000 T	0007
	THE FIRST ROW OF INFO IS USED FOR OTHER PURPOSES. THE	01025000 T	0007
	RESERVED WORDS OCCUPY THE SECOND ROW. IT IS FILLED DURING	01026000 T	0007
	INITIALIZATION;	01027000 T	0007
COMMENT	INFO FORMAT	01028000 T	0007
	FOLLOWING IS A DESCRIPTION OF THE FORMAT OF ALL TYPES OF ENTRIES	01029000 T	0007
	ENTERED IN INFO:	01030000 T	0007
	THE FIRST WORD OF ALL ENTRIES IS THE ELBAT WORD.	01031000 T	0007
	THE INCR FIELD ([27:8]) CONTAINS AN INCREMENT WHICH WHEN	01032000 T	0007
	ADDED TO THE CURRENT INDEX INTO INFO YIELDS AN INDEX TO ANY	01033000 T	0007
	ADDITIONAL INFO (IF ANY) FOR THIS ENTRY.	01034000 T	0007
	E.G. IF THE INDEX IS IX THEN INFO[(IX+INCR).LINKR,(IX+INCR).	01035000 T	0007
	LINKC] WILL CONTAIN THE FIRST WORD OF ADDITIONAL INFO.	01036000 T	0007
	THE LINK FIELD OF THE ELBAT WORD IN INFO IS DIFFERENT FROM	01037000 T	0007
	THAT OF THE ENTRY IN ELBAT PUT IN BY TABLE, THE ENTRY IN ELBAT	01038000 T	0007
	POINTS TO ITS OWN LOCATION (RELATIVE) IN INFO.	01039000 T	0007
	THE LINK IN INFO POINTS TO THE PREVIOUS ENTRY E.G., THE	01040000 T	0007
	LINK FROM STACKHEAD WHICH THE CURRENT ENTRY REPLACED.	01041000 T	0007
	FOR SIMPLICITY, I WILL CONSIDER INFO TO BE A ONE DIMENSIONAL	01042000 T	0007
	ARRAY, SO THAT THE BREAKING UP OF THE LINKS INTO ROW AND COLUMN	01043000 T	0007
	WILL NOT DETRACT FROM THE DISCUSSION.	01044000 T	0007
	ASSUME THAT THREE IDENTIFIERS A,B, AND C "SCRAMBLE" INTO	01045000 T	0007
	THE SAME STACKHEAD LOCATION IN THE ORDER OF APPEARANCE.	01046000 T	0007
	FURTHER ASSUME THERE ARE NO OTHER ENTRIES CONNECTED TO	01047000 T	0007
	THIS STACKHEAD INDEX. LET THIS STACKHEAD LOCATION BE	01048000 T	0007
	S[LL]	01049000 T	0007

NOW THE DECLARATION
 BEGIN REAL A,B,C IS ENCOUNTERED
 IF THE NEXT AVAILABLE INFO SPACE IS CALLED NEXTINFO
 THEN A IS ENTERED AS FOLLOWS:(ASSUME AN ELBAT WORD T HAS BEEN
 CONSTRUCTED FOR A)
 T.LINK← S[L], (WHICH IS ZERO AT FIRST).
 INFO[NEXTINFO]←T, S[L]←NEXTINFO,
 NEXTINFO←NEXTINFO+NUMBER OF WORDS IN THIS
 ENTRY.
 NOW S[L] POINTS TO THE ENTRY FOR A IN INFO AND THE ENTRY
 ITSELF CONTAINS THE STOP FLAG ZERO.
 B IS ENTERED SIMILARLY TO A.
 NOW S[L] POINTS TO THE ENTRY FOR B AND IT POINTS TO THE
 ENTRY FOR A.
 SIMILARLY,AFTER C IS ENTERED
 S[L] POINTS TO C,WHOSE ENTRY POINTS TO B WHOSE ENTRY
 POINTS TO A.
 THE SECOND WORD OF EACH ENTRY IN INFO IS MADE UP AS FOLLOWS:
 FWDPT =[1:1],THIS TELLS WHETHER A PROCEDURE WAS DECLARED
 FORWARD,IT IS RESET AT THE TIME OF ITS ACTUAL
 FULL DECLARATION.
 PURPT =[4:8] THIS GIVES A DECREMENT WHICH GIVES THE RELATIVE
 INDEX TO THE PREVIOUS INFO ENTRY WHEN SUBTRACTED
 FROM THE CURRENT ENTRY INDEX.
 [12:6] TELLS THE NUMBER OF CHARACTERS IN THE ENTRY.(≤64)
 [18:30] CONTAINS THE FIRST FIVE ALPHA CHARACTERS OF THE ENTRY
 AND SUCCEEDING WORDS CONTAIN ALL OVERFLOW IF NEEDED.
 THESE WORDS CONTAIN 8 CHARACTERS EACH,LEFT JUSTIFIED,
 THUS,AN ENTRY FOR SYMBOL FOLLOWED BY AN ENTRY
 FOR X WOULD APPEAR AS FOLLOWS:
 INFO[I] = ELBATWRD (MADE FOR SYMBOL)
 I+1 = OP6SYMB (P DEPENDS ON PREVIOUS ENTRY)
 I+2 = L
 I+3 = ELBATWRD (MADE FOR X)
 I+4 = 031X
 THIS SHOWS THAT INFO[I-P] WOULD POINT TO THE BEGINNING OF
 THE ENTRY BEFORE SYMBOL, AND
 INFO[I+3-3] POINTS TO THE ENTRY FOR SYMBOL.
 ALL ENTRIES OF IDENTIFIERS HAVE THE INFORMATION DESCRIBED ABOVE
 THAT IS,THE ELBAT WORD FOLLOWED BY THE WORD CONTAINING THE FIRST
 FIVE CHARACTERS OF ALPHA,AND ANY ADDITIONAL WORDS OF ALPHA IF
 NECESSARY.
 THIS IS SUFFICIENT FOR ENTRIES OF THE FOLLOWING TYPES,
 REAL
 BOOLEAN
 INTEGER
 ALPHA
 FILE
 FORMAT
 LIST
 OTHER ENTRIES REQUIRE ADDITIONAL INFORMATION.
 ARRAYS:
 THE FIRST WORD OF ADDITIONAL INFO CONTAINS THE NUMBER OF
 DIMENSIONS(IN THE LOW ORDER PART),[40:8]
 EACH SUCCEEDING WORD CONTAINS INFORMATION ABOUT EACH LOWER
 BOUND IN ORDER OF APPEARANCE,ONE WORD FOR EACH LOWER BOUND.
 THESE WORDS ARE MADE UP AS FOLLOWS:

01050000	T	0007
01051000	T	0007
01052000	T	0007
01053000	T	0007
01054000	T	0007
01055000	T	0007
01056000	T	0007
01057000	T	0007
01058000	T	0007
01059000	T	0007
01060000	T	0007
01061000	T	0007
01062000	T	0007
01063000	T	0007
01064000	T	0007
01065000	T	0007
01066000	T	0007
01067000	T	0007
01068000	T	0007
01069000	T	0007
01070000	T	0007
01071000	T	0007
01072000	T	0007
01073000	T	0007
01074000	T	0007
01075000	T	0007
01076000	T	0007
01077000	T	0007
01078000	T	0007
01079000	T	0007
01080000	T	0007
01081000	T	0007
01082000	T	0007
01083000	T	0007
01084000	T	0007
01085000	T	0007
01086000	T	0007
01087000	T	0007
01088000	T	0007
01089000	T	0007
01090000	T	0007
01091000	T	0007
01092000	T	0007
01093000	T	0007
01094000	T	0007
01095000	T	0007
01096000	T	0007
01097000	T	0007
01098000	T	0007
01099000	T	0007
01100000	T	0007
01101000	T	0007
01102000	T	0007
01103000	T	0007
01104000	T	0007
01105000	T	0007
01106000	T	0007

[23:12] = ADD OPERATOR SYLLABLE (0101) OR
SUB OPERATOR SYLLABLE (0301) CORRESPONDING
RESPECTIVELY TO WHETHER THE LOWER BOUND IS
TO BE ADDED TO THE SUBSCRIPT IN INDEXING OR
SUBTRACTED.

[35:11] = 11 BIT ADDRESS OF LOWER BOUND, IF THE LOWER BOUND
REQUIRES A PRT OR STACK CELL, OTHERWISE THE BIT
35 IS IGNORED AND THE NEXT TEN BITS ([36:10])
REPRESENT THE ACTUAL VALUE OF THE LOWER BOUND

[46:12] = 00 OR 10 DEPENDING ON WHETHER THE [35:11] VALUE
IS A LITERAL OR OPERAND, RESPECTIVELY.

PROCEDURES:

THE FIRST WORD OF ADDITIONAL INFO CONTAINS THE NUMBER OF
PARAMETERS [40:8]

IF A STREAM PROCEDURE THEN THIS WORD CONTAINS ALSO IN

[13:11] ENDING PRT ADDRESS FOR LABELS,

[7:6] NO OF LABELS REQUIRING PRT ADDRESSES, AND [1:6] NUMBER
OF LOCALS.

SUCCEEDING WORDS (ONE FOR EACH FORMAL PARAMETER, IN ORDER
OF APPEARANCE IN FORMAL PARAPART) ARE

ELBAT WORDS SPECIFYING TYPE OF EACH PARAMETER AND WHETHER
VALUE OR NOT ([10:1]).

THE ADDRESS ([16:11]) IS THE F- ADDRESS FOR EACH.

IF THE PARAMETER IS AN ARRAY THEN THE INCR FIELD ([27:8])

CONTAINS THE NUMBER OF DIMENSIONS, OTHERWISE INCR IS MEANINGLESS.

LINK ([35:13]) IS MEANINGLESS.

IF A STREAM PROCEDURE THEN THE CLASS OF EACH PARAMETER IS

THAT OF LOCAL ID OR FILE ID, DEPENDING ON WHETHER OR NOT A RELEASE
IS DONE IN THE STREAM PROCEDURE.

LABELS:

AT DECLARATION TIME THE ADDITIONAL INFO CONTAINS 0. THE SIGN

BIT TELLS WHETHER OR NOT THE DEFINITION POINT HAS BEEN REACHED.

IF SIGN = 0, THEN [36:12] CONTAINS AN ADDRESS IN CODEARRAY OF A

LIST OF FORWARD REFERENCES TO THIS LABEL. THE END OF LIST FLAG IS
0. IF SIGN = 0, THEN [36:12] CONTAINS L FOR THIS LABEL.

SWITCHES:

THE FIELD [36:12] CONTAINS L FOR THE BEGINNING OF SWITCH DECLAR-
ATION. [24:12] CONTAINS L FOR FIRST SIMPLE REFERENCE TO SWITCH.
IF SWITCH IS NOT SIMPLE, IT IS MARKED FORMAL. HERE SIMPLE MEANS
NO POSSIBILITY OF JUMPING OUT OF A BLOCK. ;

```

DEFINE MON    = [ 1: 1]#,
CLASS        = [ 2: 7]#,
FORMAL       = [ 9: 1]#,
VO          = [10: 1]#,
LVL         = [11: 5]#,
ADDRESS     = [16:11]#,
INCR        = [27: 8]#,
LINK        = [35:13]#,
LINKR       = [35: 5]#,
LINKC       = [40: 8]#;

```

COMMENT THESE DEFINES ARE USED TO PICK APART THE ELBAT WORD.

MON IS THE BIT WHICH IS ON IF THE QUANTITY IS MONITORED.
CLASS IS THE PRINCIPAL IDENTIFICATION OF A GIVEN
QUANTITY.

FORMAL IS THE BIT WHICH IS ON IF THE QUANTITY IS A FORMAL
PARAMETER.

VO IS THE VALUE-OWN BIT. IF FORMAL = 1 THEN THE BIT

```

01107000 T 0007
01108000 T 0007
01109000 T 0007
01110000 T 0007
01111000 T 0007
01112000 T 0007
01113000 T 0007
01114000 T 0007
01115000 T 0007
01116000 T 0007
01117000 T 0007
01118000 T 0007
01119000 T 0007
01120000 T 0007
01121000 T 0007
01122000 T 0007
01123000 T 0007
01124000 T 0007
01125000 T 0007
01126000 T 0007
01127000 T 0007
01128000 T 0007
01129000 T 0007
01130000 T 0007
01131000 T 0007
01132000 T 0007
01133000 T 0007
01134000 T 0007
01135000 T 0007
01136000 T 0007
01137000 T 0007
01138000 T 0007
01139000 T 0007
01140000 T 0007
01141000 T 0007
01142000 T 0007
01143000 T 0007
01144000 T 0007
01145000 T 0007
01146000 T 0007
01147000 T 0007
01148000 T 0007
01149000 T 0007
01150000 T 0007
01151000 T 0007
01152000 T 0007
01153000 T 0007
01154000 T 0007
01155000 T 0007
01156000 T 0007
01157000 T 0007
01158000 T 0007
01159000 T 0007
01160000 T 0007
01161000 T 0007
01162000 T 0007
01163000 T 0007

```

DISTINGUISHES VALUE PARAMETERS FROM OTHERS. IF
 FORMAL = 0 THEN THE BIT DISTINGUISHES OWN VARIABLES
 FROM OTHERS.
 LVL GIVES THE LEVEL AT WHICH A QUANTITY WAS DECLARED.
 ADDRESS GIVES THE STACK OR PRT ADDRESS.
 INCR GIVES A RELATIVE LINK TO ANY ADDITIONAL INFORMATION
 NEEDED, RELATIVE TO THE LOCATION IN INFO.
 LINK CONTAINS A LINK TO THE LOCATION IN INFO IF THE
 QUANTITY LIES IN ELBAT, OTHERWISE IT LINKS TO THE
 NEXT ITEM IN THE STACK. ZERO IS AN END FLAG.
 LINKR AND LINKC ARE SUBDIVISIONS OF LINK.;
 COMMENT CLASSES FOR ALL QUANTITIES - OCTAL CLASS IS IN COMMENT;
 COMMENT CLASSES FOR IDENTIFIERS;
 DEFINE UNKNOWNID =00#, COMMENT 000;
 STLABID =01#, COMMENT 001;
 LOCLID =02#, COMMENT 002;
 DEFINEDID =03#, COMMENT 003;
 LISTID =04#, COMMENT 004;
 FRMTID =05#, COMMENT 005;
 SUPERFRMTID =06#, COMMENT 006;
 REALSUBID =07#, COMMENT 007;
 SUBID =08#, COMMENT 010;
 SWITCHID =09#, COMMENT 011;
 PROCID =10#, COMMENT 012;
 INTRNSICPROCID =11#, COMMENT 013;
 STRPROCID =12#, COMMENT 014;
 BOOSTRPROCID =13#, COMMENT 015;
 REALSTRPROCID =14#, COMMENT 016;
 ALFASTRPROCID =15#, COMMENT 017;
 INTSTRPROCID =15#, COMMENT 017;
 BOOPROCID =17#, COMMENT 021;
 REALPROCID =18#, COMMENT 022;
 ALFAPROCID =19#, COMMENT 023;
 INTPROCID =19#, COMMENT 023;
 BOOID =21#, COMMENT 025;
 REALID =22#, COMMENT 026;
 ALFAID =23#, COMMENT 027;
 INTID =23#, COMMENT 027;
 BOOARRAYID =25#, COMMENT 031;
 REALARRAYID =26#, COMMENT 032;
 ALFAARRAYID =27#, COMMENT 033;
 INTARRAYID =27#, COMMENT 033;
 NAMEID =30#, COMMENT 036;
 INTNAMEID =31#, COMMENT 037;
 LABELID =32#, COMMENT 040;
 COMMENT CLASSES FOR PRIMARY BEGINNERS;
 TRUTHV =33#, COMMENT 041;
 NONLITNO =34#, COMMENT 042;
 LITNO =35#, COMMENT 043;
 STRNGCON =36#, COMMENT 044;
 LEFTPAREN =37#, COMMENT 045;
 POLISHV =38#, COMMENT 046;
 ASTRISK =39#, COMMENT 047;
 COMMENT CLASS FOR ALL DECLARATORS;
 DECLARATORS =40#, COMMENT 050;
 COMMENT CLASSES FOR STATEMENT BEGINNERS
 DOUBLEV =42#, COMMENT 052;

01164000 T 0007
 01165000 T 0007
 01166000 T 0007
 01167000 T 0007
 01168000 T 0007
 01169000 T 0007
 01170000 T 0007
 01171000 T 0007
 01172000 T 0007
 01173000 T 0007
 01174000 T 0007
 01175000 T 0007
 01176000 T 0007
 01177000 T 0007
 01178000 T 0007
 01179000 T 0007
 01180000 T 0007
 01181000 T 0007
 01182000 T 0007
 01183000 T 0007
 01184000 T 0007
 01185000 T 0007
 01186000 T 0007
 01187000 T 0007
 01188000 T 0007
 01189000 T 0007
 01190000 T 0007
 01191000 T 0007
 01192000 T 0007
 01193000 T 0007
 01194000 T 0007
 01195000 T 0007
 01196000 T 0007
 01197000 T 0007
 01198000 T 0007
 01199000 T 0007
 01200000 T 0007
 01201000 T 0007
 01202000 T 0007
 01203000 T 0007
 01204000 T 0007
 01205000 T 0007
 01205200 T 0007
 01205400 T 0007
 01206000 T 0007
 01207000 T 0007
 01208000 T 0007
 01209000 T 0007
 01210000 T 0007
 01211000 T 0007
 01212000 T 0007
 01212100 T 0007
 01212200 T 0007
 01213000 T 0007
 01214000 T 0007
 01215000 T 0007
 01222000 T 0007

FORV	=43#	COMMENT 053;
WHILEV	=44#	COMMENT 054;
DOV	=45#	COMMENT 055;
UNTILV	=46#	COMMENT 056;
ELSEV	=47#	COMMENT 057;
ENDV	=48#	COMMENT 060;
SEMICOLON	=50#	COMMENT 062;
IFV	=51#	COMMENT 063;
GOV	=52#	COMMENT 064;
IOCLASS	=53#	COMMENT 065;
BEGINV	=54#	COMMENT 066;
COMMENT CLASSES FOR STREAM RESERVED WORDS;		
SIV	=55#	COMMENT 067;
DIQ	=56#	COMMENT 070;
CIV	=57#	COMMENT 071;
TALLYV	=58#	COMMENT 072;
DSV	=59#	COMMENT 073;
SKIPV	=60#	COMMENT 074;
JUMPV	=61#	COMMENT 075;
DBV	=62#	COMMENT 076;
SBV	=63#	COMMENT 077;
TOGGLEV	=64#	COMMENT 100;
SCV	=65#	COMMENT 101;
LOCV	=66#	COMMENT 102;
DCV	=67#	COMMENT 103;
LOCALV	=68#	COMMENT 104;
LITV	=69#	COMMENT 105;
TRANSFER	=70#	COMMENT 106;
COMMENT CLASSES FOR VARIOUS MISCELLANEOUS QUANTITIES;		
COMMENTV	=71#	COMMENT 107;
FORWARDV	=72#	COMMENT 110;
STEPV	=73#	COMMENT 111;
THENV	=74#	COMMENT 112;
TOV	=75#	COMMENT 113;
VALUEV	=76#	COMMENT 114;
WITHV	=77#	COMMENT 115;
COLON	=78#	COMMENT 116;
COMMA	=79#	COMMENT 117;
CROSSHATCH	=80#	COMMENT 120;
LFTBRKET	=81#	COMMENT 121;
PERIOD	=82#	COMMENT 122;
RTBRKET	=83#	COMMENT 123;
RTPAREN	=84#	COMMENT 124;
AMPERSAND	=85#	COMMENT 125;
COMMENT CLASSES FOR OPERATORS;		
HEXOP	=86#	COMMENT 126;
BITOP	=87#	COMMENT 127;
ISOLATE	=88#	COMMENT 130;
OPERATOR	=89#	COMMENT 131;
NOTOP	=90#	COMMENT 132;
ASSIGNOP	=91#	COMMENT 133;
EQVOP	=92#	COMMENT 134;
OROP	=93#	COMMENT 135;
ANDOP	=94#	COMMENT 136;
RELOP	=95#	COMMENT 137;
ADOP	=96#	COMMENT 140;
MULOP	=97#	COMMENT 141;

01223000	T	0007
01224000	T	0007
01225000	T	0007
01226000	T	0007
01227000	T	0007
01228000	T	0007
01230000	T	0007
01231000	T	0007
01232000	T	0007
01233000	T	0007
01234000	T	0007
01235000	T	0007
01236000	T	0007
01237000	T	0007
01238000	T	0007
01239000	T	0007
01240000	T	0007
01241000	T	0007
01242000	T	0007
01243000	T	0007
01244000	T	0007
01245000	T	0007
01246000	T	0007
01247000	T	0007
01248000	T	0007
01249000	T	0007
01250000	T	0007
01251000	T	0007
01252000	T	0007
01253000	T	0007
01254000	T	0007
01255000	T	0007
01256000	T	0007
01257000	T	0007
01258000	T	0007
01259000	T	0007
01260000	T	0007
01261000	T	0007
01262000	T	0007
01263000	T	0007
01264000	T	0007
01265000	T	0007
01266000	T	0007
01266500	T	0007
01267000	T	0007
01268000	T	0007
01269000	T	0007
01270000	T	0007
01271000	T	0007
01272000	T	0007
01273000	T	0007
01274000	T	0007
01275000	T	0007
01276000	T	0007
01277000	T	0007
01278000	T	0007
01278500	T	0007

%	STRING	=99#;	COMMENT 143;	01278600	T	0007
	COMMENT	SUBCLASSES FOR DECLARATORS (KEPT IN ADDRESS);		01279000	T	0007
	QWNV	=01#;	COMMENT 01;	01280000	T	0007
	SAVEV	=02#;	COMMENT 02;	01281000	T	0007
	BOOV	=03#;	COMMENT 03;	01282000	T	0007
	REALV	=04#;	COMMENT 04;	01283000	T	0007
	ALFAV	=05#;	COMMENT 05;	01284000	T	0007
	INTV	=05#;	COMMENT 05;	01285000	T	0007
	LABELV	=07#;	COMMENT 07;	01286000	T	0007
	DUMPV	=08#;	COMMENT 10;	01287000	T	0007
	SUBV	=09#;	COMMENT 11;	01288000	T	0007
	OUTV	=10#;	COMMENT 12;	01289000	T	0007
	INV	=11#;	COMMENT 13;	01290000	T	0007
	MONITORV	=12#;	COMMENT 14;	01291000	T	0007
	SWITCHV	=13#;	COMMENT 15;	01292000	T	0007
	PROCV	=14#;	COMMENT 16;	01293000	T	0007
	ARRAYV	=15#;	COMMENT 17;	01294000	T	0007
	NAMEV	=16#;	COMMENT 20;	01295000	T	0007
	FILEV	=17#;	COMMENT 21;	01296000	T	0007
	STREAMV	=18#;	COMMENT 22;	01297000	T	0007
	DEFINEV	=19#;	COMMENT 23;	01298000	T	0007
	DEFINE DDES	= 8#;		01299000	T	0007
	ADES	= 28#;		01299010	T	0007
	PDES	= 29#;		01299020	T	0007
	LDDES	= 30#;		01299030	T	0007
	CHAR	= 31#;		01299040	T	0007
	FACTOP	= ASTRISK#;		01299100	T	0007
	OPERATORS	= HEXOP#;		01299200	T	0007
	FILEID	= 0#;		01299300	T	0007
	MAXINTRINSIC	= 150#;	% USED IN BUILDING INTABLE @ 09414120	01299400	T	0007
	INTRINSICADR	= (MAXINTRINSIC DIV 30)#;	% RESERVES SEG FOR INTABLE	01299500	T	0007
	REAL TIME1;			01300000	T	0007
PRT(75) = TIME1	BOOLEAN ASTOG;			01300100	T	0007
PRT(76) = ASTOG	BOOLEAN SAF;			01300200	T	0007
PRT(77) = SAF	INTEGER SCRAM;			01301000	T	0007
PRT(100) = SCRAM	COMMENT SCRAM CONTAINS THE SCRAMBLE INDEX FOR THE LAST IDENTIFIER			01302000	T	0007
	OR RESERVED WORD SCANNED;			01303000	T	0007
	ALPHA ARRAY ACCUM[0:10];			01304000	T	0007
PRT(101) = ACCUM	COMMENT ACCUM HOLDS THE ALPHA AND CHARACTER COUNT OF THE LAST			01305000	T	0010
	SCANNED ITEM IN A FORM COMPATIBLE WITH ITS APPEARANCE			01306000	T	0010
	IN INFO, THAT IS ACCUM[I] = QONAAAAA, ACCUM[I], I> 1,			01307000	T	0010
	HAS ANY ADDITIONAL CHARACTERS, ACCUM[0] IS USED FOR			01308000	T	0010
	THE ELBAT WORD BY THE ENTER ROUTINES;			01309000	T	0010
	ARRAY STACKHEAD[0:125];			01310000	T	0010
PRT(102) = STACKHEAD	COMMENT STACKHEAD[N] CONTAINS AN INDEX INTO INFO GIVING THE TOP			01311000	T	0012
	ITEM IN THE N-TH STACK;			01312000	T	0012
	INTEGER COUNT;			01313000	T	0012
PRT(103) = COUNT	COMMENT COUNT CONTAINS THE NUMBER OF CHARACTORS OF THE LAST ITEM			01314000	T	0012
	SCANNED;			01315000	T	0012
	ALPHA Q;			01316000	T	0012

PRT(104) = Q	COMMENT Q CONTAINS ACCUM[1] FOR THE LAST IDENTIFIER OR RESERVED WORD SCANNED;	01317000 T	0012
	ARRAY ELBAT[0:75]; INTEGER I, NXTELBT;	01318000 T	0012
PRT(105) = ELBAT		01319000 T	0012
PRT(106) = I			
PRT(107) = NXTELBT	COMMENT ELBAT IS AN ARRAY HOLDING ELBAT WORDS FOR RECENTLY SCANNED QUANTITIES. THE TABLE ROUTINE MAINTAINS THIS ARRAY, (ELBAT IS TABLE SPELLED BACKWARDS.) THE TABLE ROUTINE GUARANTIES THAT ELBAT ALWAYS CONTAINS THE ELBAT WORDS FOR THE LAST 10 QUANTITIES SCANNED, NXTELBT IS AN INDEX POINTING TO THE NEXT AVAILABLE WORD IN ELBAT. I IS AN INDEX USED BY THE REST OF THE COMPILER TO FETCH THINGS FROM ELBAT. I IS ALSO MAINTAINED BY THE TABLE ROUTINE;	01320000 T	0015
	INTEGER ELCLASS;	01321000 T	0015
PRT(110) = ELCLASS	COMMENT ELCLASS USUALLY CONTAINS ELBAT[I].CLASS;	01322000 T	0015
	INTEGER FCR, NCR, LCR, TLCR, CLCR;	01323000 T	0015
PRT(111) = FCR		01324000 T	0015
PRT(112) = NCR		01325000 T	0015
PRT(113) = LCR		01326000 T	0015
PRT(114) = TLCR		01327000 T	0015
PRT(115) = CLCR		01328000 T	0015
	INTEGER MAXTLCR;	01329000 T	0015
PRT(116) = MAXTLCR	COMMENT FCR CONTAINS ABSOLUTE ADDRESS OF THE FIRST CHARACTER OF THE CARD IMAGE CURRENTLY BEING SCANNED, NCR THE ADDRESS OF THE NEXT CHARACTER TO BE SCANNED, AND LCR THE LAST CHARACTER (COLUMN 73), TLCR AND CLCR CONTAIN ADDRESS OF THE LAST CHARACTER IN THE TAPE AND CARD BUFFERS, MAXTLCR IS THE MAXIMUM OF TLCR WHEN THE INPUT IS BLOCKED;	01330000 T	0015
	ARRAY TEN[-46:69];	01331000 T	0015
PRT(117) = TEN		01332000 T	0015
	DEFINE PRTBASE=129#, PRTOP=896#; COMMENT PASE AND TOP OF PRT;	01333000 T	0015
	ARRAY PRT[PRTBASE:PRTOP];	01334000 T	0015
PRT(120) = PRT	INTEGER DISKADR, CORADR; COMMENT GLOBALS FOR PROGDESCBLUR;	01335000 T	0015
PRT(121) = DISKADR		01336000 T	0015
PRT(122) = CORADR		01337000 T	0015
	INTEGER SGAVL; COMMENT NEXT AVAILABLE SEGMENT NUMBER;	01340000 T	0015
PRT(123) = SGAVL		01341000 T	0017
	INTEGER SGNO; COMMENT THIS IS THE CURRENT SEGMENT NUMBER;	01342000 T	0017
PRT(124) = SGNO		01343000 T	0017
	ARRAY COP, WOP[0:127];	01344000 T	0020
PRT(125) = COP		01369000 T	0020
PRT(126) = WOP		01370000 T	0020
	COMMENT THE EMIT ROUTINES PLACE EACH SYLLABLE INTO THE EDOC ARRAY AS SPECIFIED BY "L".	01371000 T	0020
	IF THE DEBUGTOG IS TRUE COP AND WOP ARE FILLED WITH THE BCD FOR THE OPERATORS, OTHERWISE THEY ARE NOT USED;	01372000 T	0023
	REAL LASTENTRY;	01373000 T	0023
PRT(127) = LASTENTRY	COMMENT LASTENTRY IS USED BY EMITNUM AND CONSTANTCLEAN, IT POINTS INTO INFO[0,*] AT THE NEXT AVAILABLE CELL FOR CONSTANTS;	01374000 T	0023
	BOOLEAN MRCLEAN;	01375000 T	0023
		01376000 T	0023
		01377000 T	0023
		01378000 T	0023
		01379000 T	0023

PRT(130) = MRCLEAN	COMMENT NO CONSTANTCLEAN ACTION TAKES PLACE WHILE MRCLEAN IS FALSE. THIS FEATURE IS USED BY BLOCK BECAUSE OF THE POSSIBILITY THAT CONSTANTCLEAN WILL USE INFO(NEXTINFO) DURING AN ARRAY DECLARATION ;	01380000 T 0023
	REAL GT1,GT2,GT3,GT4,GT5;	01381000 T 0023
PRT(131) = GT1		01382000 T 0023
PRT(132) = GT2		01383000 T 0023
PRT(133) = GT3		01384000 T 0023
PRT(134) = GT4		
PRT(135) = GT5		
	INTEGER GTI1;	01384500 T 0023
PRT(136) = GTI1	COMMENT THESE VARIABLES ARE USED FOR TEMPORARY STORAGE;	01385000 T 0023
	INTEGER RESULT;	01386000 T 0023
PRT(137) = RESULT	COMMENT THIS VARIABLE IS USED FOR A DUAL PURPOSE BY THE TABLE ROUTINE AND THE SCANNER. THE TABLE ROUTINE USES THIS VARIABLE TO SPECIFY SCANNER OPERATIONS AND THE SCANNER USES IT TO INFORM THE TABLE ROUTINE OF THE ACTION TAKEN;	01387000 T 0023
	INTEGER LASTUSED;	01388000 T 0023
PRT(140) = LASTUSED	COMMENT LASTUSED IS A VARIABLE THAT CONTROLS THE ACTION OF READACARD, THE ROUTINE WHICH READS CARDS AND INITIALIZES OR PREPARES THE CARD FOR THE SCANNER.	01389000 T 0023
	LASTUSED LAST CARD READ FROM	01390000 T 0023
	-----	01391000 T 0023
	1 CARD READER ONLY, NO TAPE.	01392000 T 0023
	2 CARD READER, TAPE AND CARD MERGE.	01393000 T 0023
	3 TAPE, TAPE AND CARD MERGE.	01394000 T 0023
	4 INITIALIZATION ONLY, CARD ONLY.	01394500 T 0023
	;	01394600 T 0023
	BOOLEAN LINKTOG;	01395000 T 0023
PRT(141) = LINKTOG	COMMENT LINKTOG IS FALSE IF THE LAST THING EMITTED IS A LINK, OTHERWISE IT IS TRUE;	01396000 T 0023
	INTEGER LEVEL,FRSTLEVEL,SUBLEVEL,MODE;	01397000 T 0023
PRT(142) = LEVEL		01398000 T 0023
PRT(143) = FRSTLEVEL		01398300 T 0023
PRT(144) = SUBLEVEL		01399000 T 0023
PRT(145) = MODE	COMMENT THESE VARIABLES ARE MAINTAINED BY THE BLOCK ROUTINE TO KEEP TRACK OF LEVELS OF DEFINITION. LEVEL GIVES THE DEPTH OF NESTING IN DEFINITION, WHERE EACH BLOCK AND EACH PROCEDURE GIVES RISE TO A NEW LEVEL. SUBLEVEL GIVES THE LEVEL OF THE PARAMETERS OF THE PROCEDURE CURRENTLY BEING COMPILED. FRSTLEVEL IS THE LEVEL OF THE PARAMETERS OF THE MOST GLOBAL OF THE PROCEDURES CURRENTLY BEING COMPILED. MODE IS THE CURRENT DEPTH OF THE PROCEDURE IN WHICH WE ARE NESTED (AT COMPILE TIME);	01400000 T 0023
	BOOLEAN ERRORTOG;	01401000 T 0023
PRT(146) = ERRORTOG	COMMENT ERRORTOG IS TRUE IF MESSAGES ARE CURRENTLY ACCEPTABLE TO THE ERROR ROUTINES. ERRORCOUNT IS THE COUNT OF ERROR MSSGS;	01402000 T 0023
	BOOLEAN ENDTOG; COMMENT ENDTOG TELLS THE TABLE TO ALLOW	
PRT(147) = ENDTOG	COMMENT TO BE PASSED BACK TO COMPOUNDTAIL;	01403000 T 0023
		01404000 T 0023
		01405000 T 0023
		01406000 T 0023
		01407000 T 0023
		01408000 T 0023
		01409000 T 0023
		01410000 T 0023
		01411000 T 0023
		01412000 T 0023
		01413000 T 0023
		01414000 T 0023
		01415000 T 0023
		01416000 T 0023

PRT(150) = STREAMTOG	BOOLEAN STREAMTOG; COMMENT STREAMTOG IS TRUE IF WE ARE COMPILING STREAM STATEMENT. IT IS USED TO CONTROL COUMPOUNDTAIL; DEFINE FS = 1#, FP = 2#, FL = 3#, FR = 4#; COMMENT THESE DEFINES ARE USED WHEN CALLING THE VARIABLE ROUTINE. THEIR PURPOSES IS TO TELL VARIABLE WHO IS CALLING. THEIR MEANING IS: FS MEANS FROM STATEMENT, FP MEANS FROM PRIMARY, FL MEANS FROM LIST, FR MEANS FROM FOR; INTEGER L; PRT(151) = L COMMENT L IS THE LOCATION OF THE NEXT SYLLABLE TO BE EMITTED; DEFINE BLOCKCTR = 16#, JUNK = 17 #, XITR = 18 #, LSTRIN = 19#; DEFINE ATYPE = 3#, BTYPE=ATYPE#, DTYPE=ATYPE#; BOOLEAN TB1; PRT(152) = TB1 COMMENT TB1 IS A TEMPORARY BOOLEAN VARIABLE; INTEGER JUMPCTR; PRT(153) = JUMPCTR COMMENT JUMPCTR IS A VARIABLE USED FOR COMMUNICATION BETWEEN BLOCK AND GENGO. IT GIVES HIGHEST LEVEL TO WHICH A JUMP HAS BEEN MADE FROM WITHIN A THE PRESENTLY BEING COMPILED SEGMENT. THE BLOCK COMPILES CODE TO INCREMENT AND DECRE- MENT THE BLOCKCTR ON THE BASIS OF JUMPCTR AT COMPLETION OF COMPIATION OF A SEGMENT - I.E. THE BLOCKCTR IS TALLIED IF LEVEL = JUMPCTR; REAL STLB; PRT(154) = STLB COMMENT STLB IS USED BY VARIABLE AND ACTUALPARAPART TO COMMUNICATE THE LOWER BOUND INFORMATION FOR THE LAST DIMENSION OF THE ARRAY INVOLVED IN A ROW DESIGNATOR. THE FORMAT OF THE INFORMATION IS THAT OF INFO. STLB IS ALSO SOMETIMES USED FOR TEMPORARY STORAGE; DEFINE BUMPL = L+L+2#; COMMENT BUMPL IS USED MOSTLY TO PREPARE A FORWARD JUMP; DEFINE IDMAX = LABELID#; COMMENT IDMAX IS THE MAXIMUM CLASS NUMBER FOR IDENTIFIERS; INTEGER DEFINECTR, DEFINEINDEX; PRT(155) = DEFINECTR PRT(156) = DEFINEINDEX REAL JOINFO, COMMENT POINTS TO PSEUDO LABEL FOR JUMP OUTS; PRT(157) = JOINFO LPRT, COMMENT SHOWS LOCATION OF THE LAST LABEL IN THE PRT ; PRT(160) = LPRT NESTLEVEL, COMMENT COUNTS NESTING FOR GO TO AND JUMP OUTS; PRT(161) = NESTLEVEL JUMPLEVEL; COMMENT NUMBER OF LEVELS TO BE JUMPED OUT; PRT(162) = JUMPLEVEL COMMENT THE REALS ABOVE ARE FOR STREAM STATEMENT; ARRAY MACRO[0:35];	01417000 T 0023 01418000 T 0023 01419000 T 0023 01420000 T 0023 01421000 T 0023 01422000 T 0023 01423000 T 0023 01424000 T 0023 01425000 T 0023 01426000 T 0023 01427000 T 0023 01428000 T 0023 01429000 T 0023 01430000 T 0023 01452000 T 0023 01457000 T 0023 01458000 T 0023 01459000 T 0023 01460000 T 0023 01461000 T 0023 01462000 T 0023 01463000 T 0023 01464000 T 0023 01465000 T 0023 01466000 T 0023 01467000 T 0023 01468000 T 0023 01469000 T 0023 01470000 T 0023 01471000 T 0023 01472000 T 0023 01473000 T 0023 01474000 T 0023 01475000 T 0023 01476000 T 0023 01477000 T 0023 01478000 T 0023 01479000 T 0023 01480000 T 0023 01481000 T 0023 01482000 T 0023 01483000 T 0023 01484000 T 0023 01485000 T 0023 01486000 T 0023 01487000 T 0023
----------------------	---	--

PRT(163) = MACRO	COMMENT MACRO IS FILLED WITH SYLLABLES FOR STREAM STATEMENT;	01488000 T	0026
	REAL P, COMMENT CONTAINS NUMBER OF FORMALS FOR STREAM PROCS;	01489000 T	0026
PRT(164) = P			
	Z; COMMENT CONTAINS 1ST WORD OF INFO FOR STREAM FUNCTIONS;	01490000 T	0026
PRT(165) = Z			
	ARRAY NEWTAPBUF[019];	01490510 T	0026
PRT(166) = NEWTAPBUF			
	SAVE ARRAY DEFINEARRAY[0123];	01491000 T	0028
PRT(167) = DEFINEARRAY			
	COMMENT THESE VARIABLES ARE USED TO CONTROL ACTION OF THE DEFINE.	01492000 T	0031
	DEFINCTR COUNTS DEPTH OF NESTING OF DEFINE=# PAIRS.	01493000 T	0031
	THE CROSSHATCH PART OF THE TABLE ROUTINE USES DEFINCTR	01494000 T	0031
	TO DETERMINE THE MEANING OF A CROSSHATCH. DEFINEINDEX IS	01495000 T	0031
	THE NEXT AVAILABLE CELL IN THE DEFINEARRAY. THE DEFINE=	01496000 T	0031
	ARRAY HOLDS THE ALPHA OF THE DEFINE BEING RECREATED AND	01497000 T	0031
	THE PREVIOUS VALUES OF LASTUSED, LCR, AND NCR;	01498000 T	0031
	INTEGER BEGINCTR;	01499000 T	0031
PRT(170) = BEGINCTR			
	COMMENT BEGINCTR GIVES THE NUMBER OF UNMATCHED BEGINS. IT IS USED	01500000 T	0031
	FOR ERROR CONTROL ONLY;	01501000 T	0031
	INTEGER DIALA,DIALB;	01502000 T	0031
PRT(171) = DIALA			
PRT(172) = DIALB			
	COMMENT THESE VARIABLES GIVE THE LAST VALUE TO WHICH A AND B WERE	01503000 T	0031
	DIALED. THIS GIVES SOME LOCAL OPTIMIZATION. EMITD	01504000 T	0031
	WORRIES ABOUT THIS. OTHER ROUTINES CAUSE A LOSS OF MEMORY	01505000 T	0031
	BY SETTING DIALA AND DIALB TO ZERO;	01506000 T	0031
	BOOLEAN RRB1; COMMENT RRB1---RRBN ARE BOOLEAN VARIABLES THAT SERVE THE	01522000 T	0031
PRT(173) = RRB1			
	SAME FUNCTION AS RR1---RRN FOR REAL VARIABLES. SEE	01523000 T	0031
	COMMENT AT RR1;	01524000 T	0031
	BOOLEAN RRB2; COMMENT SEE COMMENT AT RRB1 DECLARATION;	01525000 T	0031
PRT(174) = RRB2			
	DEFINE ARRAYMONFILE = [27:11]; COMMENT ARRAYMONFILE IS THE DEFINE FOR	01526000 T	0031
	THE ADDRESS OF THE FILE DESCRIPTOR IN	01527000 T	0031
	THE FIRST WORD OF ADDITIONAL INFO;	01528000 T	0031
	DEFINE SVARMONFILE = [37:11]; COMMENT MONITORFILE IS THE DEFINE FOR	01529000 T	0031
	THE ADDRESS OF THE FILE DESCRIPTOR IN	01530000 T	0031
	INFO FOR MONITORED SIMPLE VARIABLES;	01531000 T	0031
	DEFINE NODIMPART = [40:8]; COMMENT THE FIRST ADDITIONAL WORD OF INFO	01532000 T	0031
	FOR ARRAYS CONTAINS THE NUMBER OF DIMENSIONS	01533000 T	0031
	IN NODIMPART;	01534000 T	0031
	DEFINE LABLMONFILE = [13:11]; COMMENT LABLMONFILE DESIGNATES THE BIT	01535000 T	0031
	POSITION IN THE FIRST WORD OF ADDITIONAL	01536000 T	0031
	INFO THAT CONTAINS THE MONITOR FILE	01537000 T	0031
	ADDRESS FOR LABELS;	01538000 T	0031
	DEFINE SWITMONFILE = [13:11]; COMMENT SWITMONFILE DESIGNATES THE BIT	01539000 T	0031
	POSITION IN THE FIRST WORD OF ADDITIONAL	01540000 T	0031
	INFO THAT CONTAINS THE MONITOR FILE	01541000 T	0031
	ADDRESS FOR LABELS;	01542000 T	0031
	DEFINE FUNCMONFILE = [27:11]; COMMENT FUNCMONFILE DESIGNATES THE BIT	01543000 T	0031
	POSITION IN THE FIRST WORD OF ADDITIONAL	01544000 T	0031
	INFO THAT CONTAINS THE MONITOR FILE	01545000 T	0031
	ADDRESS FOR LABELS;	01546000 T	0031
	DEFINE DUMPEE = [2:11]; COMMENT THE DUMPEE FIELD IN THE FIRST	01547000 T	0031
	ADDITIONAL WORD OF INFO FOR LABELS CONTAINS	01548000 T	0031

	THE ADDRESS OF THE COUNTER THAT IS INCREMENTED	01549000 T	0031
	EACH TIME THE LABEL IS PASSED IF THAT LABEL	01550000 T	0031
	APPEARS IN A DUMP DECLARATION;	01551000 T	0031
DEFINE DUMPOR = [24:11]#;	COMMENT THE DUMPOR FIELD IN THE FIRST	01552000 T	0031
	ADDITIONAL WORD OF INFO FOR LABELS CONTAINS	01553000 T	0031
	THE ADDRESS OF THE ROUTINE THAT IS GENERATED	01554000 T	0031
	FROM THE DUMP DECLARATION THAT IN TURN CALLS	01555000 T	0031
	THE PRINTI ROUTINE;	01556000 T	0031
DEFINE SUBOP=48#;		01556500 T	0031
FILE OUT CODE DISK SERIAL[1:1](1,1023);		01556900 T	0031
PRT(175) = CODE	FILE IN CARD(RR1,10,RR2);	01557000 T	0035
PRT(176) = CARD	FILE OUT LINE DISK SERIAL[20:2400](RR3,15,RR4,SAVE 10);	01558000 T	0039
PRT(177) = LINE			
PRT(200) =			
PRT(201) = FILE	ATTRBUTS		
	ARRAY LIN[0:20]; COMMENT PRINT OUTPUT BUILT IN LIN;	01559010 T	0046
PRT(202) = LIN	INTEGER DA;	01559020 T	0049
PRT(203) = DA	SAVE FILE OUT NEWTAPE DISK SERIAL[20:2400](RR5,RR6,RR7,SAVE 1);	01560000 T	0049
PRT(204) = NEWTAPE	FILE IN TAPE "OCDIMG"(2,RR8,RR9);	01561000 T	0056
PRT(205) = TAPE	SAVE ARRAY CBUF,TBUF[0:9]; % INPUT BUFFERS,	01561056 T	0060
PRT(206) = CBUF			
PRT(207) = TBUF	FILE OUT CODISK DISK SERIAL [20:600] (2,30,300);	01561300 T	0064
PRT(210) = CODISK	FILE OUT DISK DISK [1:2100] "MCP" "DISK"(3,30,300,SAVE 99);	01561400 T	0068
PRT(211) = DISK	DEFINE MCPTYPE = 63#,	01561410 T	0075
	DCINTYPE = 62#,	01561420 T	0075
	TSSINTYPE = 61#;	01561430 T	0075
	COMMENT ESPOL CODE FILES ARE UNIQUELY TYPED IN THEIR FILE	01561440 T	0075
	HEADERS. HEADER[4],[36:6] IS THE FIELD USED TO CONTAIN	01561450 T	0075
	THE TYPE;	01561460 T	0075
	FILE OUT DECK 0 (2,10);	01561500 T	0075
PRT(212) = DECK	FILE STUFF DISK SERIAL[20:150](2,10,30,SAVE 15);	01561600 T	0079
PRT(213) = STUFF	ARRAY TWXA[0:16];	01561700 T	0086
PRT(214) = TWXA	REAL C;	01562000 T	0089
PRT(215) = C	COMMENT C CONTAINS ACTUAL VALUE OF LAST CONSTANT SCANNED;	01563000 T	0089
	REAL T;	01564000 T	0089
PRT(216) = T	COMMENT T IS A TEMPORARY CELL;	01565000 T	0089
	INTEGER TCOUNT;	01566000 T	0089
PRT(217) = TCOUNT	REAL STACKCT;	01566010 T	0089
PRT(220) = STACKCT	COMMENT TCOUNT IS A VARIABLE WHICH HOLDS A PREVIOUS VALUE OF COUNT	01567000 T	0089
	FOR THE USE OF CONVERT;	01568000 T	0089
	DEFINE LASTSEQUENCE = 145#;	01569000 T	0089

LASTSEQROW = 2#;

```

                                REAL FOULED;
PRT(221) = FOULED

                                BOOLEAN
                                FUNCTOG, COMMENT TELLS WHETHER PROCEDURE BEING DECLARED IS A
PRT(222) = FUNCTOG                                FUNCTION;

                                P2, COMMENT GENERALLY TELLS WHETHER OWN WAS SEEN;
PRT(223) = P2

                                P3, COMMENT TELLS WHETHER SAVE WAS SEEN;
PRT(224) = P3

                                VONF, COMMENT VALUE OR OWN FIELD OF ELBAT WORD;
PRT(225) = VONF

                                FORMALF, COMMENT FORMAL FIELD OF ELBAT WORD;
PRT(226) = FORMALF

                                PTOG, COMMENT TELLS THAT FORMAL PARAPART IS BEING PROCESSED;
PRT(227) = PTOG

                                SPECTOG,
PRT(230) = SPECTOG

                                STOPENTRY, COMMENT THIS MAKES THE ENTRY PROCEDURE ENTER ONLY
PRT(231) = STOPENTRY                                ONE ID AND THEN EIXT;

                                AJUMP; COMMENT TELLS WHETHER A JUMP IS HANGING;
PRT(232) = AJUMP

                                BOOLEAN STOPDEFINE;
PRT(233) = STOPDEFINE

                                INTEGER MAXSAVE;
PRT(234) = MAXSAVE

                                COMMENT THIS CONTAINS THE SIZE OF THE MAXIMUM SAVE ARRAY
                                DECLARED. IT IS USED TO HELP DETERMINE STORAGE REQUIREMENTS
                                FOR THE PROGRAM PARAMETER BLOCK FOR THE OBJECT PROGRAM;

                                REAL
                                KLASSF, COMMENT CLASS IN LOW ORDER 7 BITS;
PRT(235) = KLASSF

                                ADDR5F, COMMENT ADDRESS IN LOW ORDER 11 BITS;
PRT(236) = ADDR5F

                                LEVELF, COMMENT LVL IN LOW ORDER 5 BITS;
PRT(237) = LEVELF

                                LINKF, COMMENT LINK IN LOW ORDER 13 BITS;
PRT(240) = LINKF

                                INCRF, COMMENT INCR CN LOW ORDER 8 BITS;
PRT(241) = INCRF

                                PROINFO, COMMENT CONTAINS ELBAT WORD FOR PROCEDURE BEING

```

```

01570000 T 0089
01571000 T 0089
01572000 T 0089
01573000 T 0089
01574000 T 0089
01575000 T 0089
01576000 T 0089
01577000 T 0089
01578000 T 0089
01579000 T 0089
01580000 T 0089
01581000 T 0089
01582000 T 0089
01583000 T 0089
01583100 T 0089

01584000 T 0089
01585000 T 0089
01586000 T 0089

01587000 T 0089
01588000 T 0089

01589000 T 0089

01590000 T 0089

01591000 T 0089

01592000 T 0089

01593000 T 0089

01594000 T 0089

01595000 T 0089
01596000 T 0089

01597000 T 0089

01598000 T 0089

01599000 T 0089
01600000 T 0089
01601000 T 0089
01602000 T 0089
01603000 T 0089

01604000 T 0089

01605000 T 0089

01606000 T 0089

01607000 T 0089

01608000 T 0089

```


PRT(242) = PROINFO		DECLARED;	01609000 T 0089
	G,	COMMENT GLOBAL TEMPORARY FOR BLOCK;	01610000 T 0089
PRT(243) = G			
	TYPEV,	COMMENT USED TO CARRY CLASS OF IDENTIFIER	01611000 T 0089
PRT(244) = TYPEV			
	PROADO,	BEING DECLARED;	01612000 T 0089
PRT(245) = PROADO		COMMENT CONTAINS ADDRESS OF PROCEDURE BEING	01613000 T 0089
	MARK ,	DECLARED;	01614000 T 0089
PRT(246) = MARK		COMMENT CONTAINS INDEX INTO INFO WHERE FIRST WORD	01615000 T 0089
	PJ,	OF ADDITIONAL INFO FOR A PROCEDURE ENTRY;	01616000 T 0089
PRT(247) = PJ		COMMENT FORMAL PARAMETER COUNTER;	01617000 T 0089
	J,	COMMENT ARRAY COUNTER;	01618000 T 0089
PRT(250) = J			
	LASTINFO,	COMMENT INDEX TO LAST ENTRY IN INFO;	01619000 T 0089
PRT(251) = LASTINFO			
	NEXTINFO,	COMMENT INDEX FOR NEXT ENTRY IN INFO;	01620000 T 0089
PRT(252) = NEXTINFO			
	FIRSTX,	COMMENT RELATIVE ADD OF FIRST EXECUTABLE CODE	01621000 T 0089
PRT(253) = FIRSTX			
	SAVEL;	IN BLOCK, INITIALIZED TO 4095 EACH TIME;	01622000 T 0089
PRT(254) = SAVEL		COMMENT SAVE LOCATION FOR FIXUPS IN BLOCK;	01623000 T 0089
	INTEGER NCII;	COMMENT THIS CONTAINS THE COUNT OF CONSTANTS	01624000 T 0089
PRT(255) = NCII			
		ENTERED IN INFO AT ANY GIVEN TIME;	01625000 T 0089
	PROCEDURE UNHOOK; FORWARD;		01626000 T 0089
PRT(256) = UNHOOK			
	PROCEDURE MAKEUPACCUM; FORWARD;		01627000 T 0093
PRT(257) = MAKEUPACCUM			
	DEFINE PURPT=[418]#, SECRET=2#;		01628000 T 0093
	COMMENT THESE DEFINES GIVE THE NAMES OF THE WORD MODE OPERATORS. THE		01629000 T 0093
	NUMBERS REFER TO THE APPROPRIATE SECTION OF THE PRODUCT SPECS. THE		01630000 T 0093
	FULL NAME IS ALSO GIVEN;		01631000 T 0093
	DEFINE		01632000 T 0093
	ADD = 16#,	COMMENT (0101) 7.4.2.1	01633000 T 0093
	BBC = 22#,	COMMENT (0131) 7.4.5.4	01634000 T 0093
	BBW = 534#,	COMMENT (4131) 7.4.5.2	01635000 T 0093
	BFC = 38#,	COMMENT (0231) 7.4.5.3	01636000 T 0093
	BFW = 550#,	COMMENT (4231) 7.4.5.1	01637000 T 0093
	CDC = 168#,	COMMENT (1241) 7.4.10.4	01638000 T 0093
	CHS = 134#,	COMMENT (1031) 7.4.7.11	01639000 T 0093
	COC = 40#,	COMMENT (0241) 7.4.10.3	01640000 T 0093
	COM = 130#,	COMMENT (1011) 7.4.10.5	01641000 T 0093
	DEL = 10#,	COMMENT (0045) 7.4.9.3	01642000 T 0093
	DUP = 261#,	COMMENT (2025) 7.4.9.2	01643000 T 0093
	EQL = 581#,	COMMENT (4425) 7.4.4.3	01644000 T 0093
	LBC = 278#,	COMMENT (2131) 7.4.5.9	01645000 T 0093
	LBU = 790#,	COMMENT (6131) 7.4.5.7	01646000 T 0093
	GEQ = 21#,	COMMENT (0125) 7.4.4.2	01647000 T 0093
	LFC = 294#,	COMMENT (2231) 7.4.5.8	01648000 T 0093
	LFU = 806#,	COMMENT (6231) 7.4.5.6	01649000 T 0093
	GTR = 37#,	COMMENT (0225) 7.4.4.1	01650000 T 0093
	IDV = 384#,	COMMENT (3001) 7.4.2.5	01651000 T 0093
		ADD;	
		BRANCH BACKWARD CONDITIONAL;	
		BRANCH BACKWARD;	
		BRANCH FORWARD CONDITIONAL;	
		BRANCH FORWARD;	
		CONSTRUCT DESCRIPTOR CALL;	
		CHANGE SIGN;	
		CONSTRUCT OPERAND CALL;	
		COMMUNICATION OPERATOR;	
		DELETE;	
		DUPLICATE;	
		EQUAL;	
		GO BACKWARD CONDITIONAL;	
		GO BACKWARD (WORD);	
		GREATER THAN OR EQUAL TO;	
		GO FORWARD CONDITIONAL;	
		GO FORWARD (WORD);	
		GREATER THAN;	
		INTEGER DIVIDE;	

INX = 24#	COMMENT (0141) 7.4.10.2	INDEX;	01652000 T	0093
ISD = 532#	COMMENT (4121) 7.4.6.3	INTEGER STORE DESTRUCTIVE;	01653000 T	0093
ISN = 548#	COMMENT (4221) 7.4.6.4	INTEGER STORE NON-DESTRUCT;	01654000 T	0093
LEQ = 533#	COMMENT (4125) 7.4.4.4	LESS THAN OR EQUAL TO;	01655000 T	0093
LND = 67#	COMMENT (0415) 7.4.3.1	LOGICAL AND;	01656000 T	0093
LNG = 19#	COMMENT (0115) 7.4.3.4	LOGICAL NEGATE;	01657000 T	0093
LOD = 260#	COMMENT (2021) 7.4.10.1	LOAD OPERATOR;	01658000 T	0093
LOR = 35#	COMMENT (0215) 7.4.3.2	LOGICAL OR;	01659000 T	0093
LQV = 131#	COMMENT (1015) 7.4.3.3	LOGICAL EQUIVALENCE;	01660000 T	0093
LSS = 549#	COMMENT (4225) 7.4.4.5	LESS THAN;	01661000 T	0093
MKS = 72#	COMMENT (0441) 7.4.8.1	MARK STACK;	01662000 T	0093
MUL = 64#	COMMENT (0401) 7.4.2.3	MULTIPLY;	01663000 T	0093
NEQ = 69#	COMMENT (0425) 7.4.4.6	NOT EQUAL TO;	01664000 T	0093
NOP = 11#	COMMENT (0055) 7.4.7.1	NO OPERATION;	01665000 T	0093
PRL = 18#	COMMENT (0111) 7.4.10.6	PROGRAM RELEASE;	01666000 T	0093
PRTE = 12#	COMMENT (0061) 7.4.10.9	EXTEND PRT;	01667000 T	0093
RDV = 896#	COMMENT (7001) 7.4.2.6	REMAINDER DIVIDE;	01668000 T	0093
RTN = 39#	COMMENT (0235) 7.4.8.3	RETURN NORMAL;	01669000 T	0093
RTS = 167#	COMMENT (1235) 7.4.8.4	RETURN SPECIAL;	01670000 T	0093
SND = 132#	COMMENT (1021) 7.4.6.2	STORE NON-DESTRUCTIVE;	01671000 T	0093
SSP = 582#	COMMENT (4431) 7.4.7.10	SET SIGN PLUS;	01672000 T	0093
STD = 68#	COMMENT (0421) 7.4.6.1	STORE DESTRUCTIVE;	01673000 T	0093
SUB = 48#	COMMENT (0301) 7.4.2.2	SUBTRACT;	01674000 T	0093
XCH = 133#	COMMENT (1025) 7.4.9.1	EXCHANGE;	01675000 T	0093
XIT = 71#	COMMENT (0435) 7.4.8.2	EXIT;	01676000 T	0093
ZP1 = 322#	COMMENT (2411) 7.4.10.8	CONDITIONAL HALT;	01677000 T	0093
SC1 = 1003#	COMMENT (7655)	SCAN OUT INITIALIZE;	01677050 T	0093
SAN = 1004#	COMMENT (7661)	SYSTEM ATTENTION NEEDED;	01677100 T	0093
SCS = 1019#	COMMENT (7755)	SCAN OUT STOP;	01677150 T	0093
COMMENT THESE DEFINES ARE USED BY EMITD;			01678000 T	0093
DEFINE			01679000 T	0093
DIA = 45#	COMMENT (XX55) 7.4.7.1	DIAL A;	01680000 T	0093
DIB = 49#	COMMENT (XX61) 7.4.7.2	DIAL B;	01681000 T	0093
TRB = 53#	COMMENT (XX65) 7.4.7.3	TRANSFER BITS;	01682000 T	0093
REAL MAXSTACK,STACKCTR;			01683000 T	0093
PRT(260) = MAXSTACK				
PRT(261) = STACKCTR				
INTEGER MAXROW;			01684000 T	0093
PRT(262) = MAXROW				
COMMENT THIS CONTAINS THE MAXIMUM ROW SIZE OF ALL NON-SAVE ARRAYS DECLARED. ITS USE IS LIKE THAT OF MAXSAVE;			01685000 T	0093
INTEGER SEGSIZEMAX; COMMENT CONTAINS MAX SEGMENT SIZE;			01686000 T	0093
PRT(263) = SEGSIZEMAX			01687000 T	0093
INTEGER F;			01688000 T	0093
PRT(264) = F				
REAL NLO,NHI,TLO,THI;			01689000 T	0093
PRT(265) = NLO				
PRT(266) = NHI				
PRT(267) = TLO				
PRT(270) = THI				
BOOLEAN DPTOG;			01690000 T	0093
PRT(271) = DPTOG				
COMMENT THE ABOVE THINGS ARE TEMP STORAGE FOR DOUBLE NOS;			01691000 T	0093
BOOLEAN DOLLAR2TOG;			01691500 T	0093
PRT(272) = DOLLAR2TOG				
DEFINE FZERO=896#;			01692000 T	0093
REAL T1,T2,N,K,AKKUM;			01693000 T	0093

PRT(273) = T1			
PRT(274) = T2			
PRT(275) = N			
PRT(276) = K			
PRT(277) = AKKUM			
PRT(300) = STOPGSP	BOOLEAN STOPGSP;	01694000 T	0093
PRT(301) = BUP	INTEGER BUP;	01695000 T	0093
PRT(302) = INLINETO	BOOLEAN INLINETO;	01695500 T	0093
	COMMENT UNIQUE GLOBAL TEMP FOR BLOCK;	01696000 T	0093
	ARRAY GTA1[0:10];	01697000 T	0093
PRT(303) = GTA1	BOOLEAN ARRAY SPRT[0:31];	01698000 T	0095
PRT(304) = SPRT	COMMENT SPRT IS TO BE CONSIDERED TO BE AN ARRAY OF 32 32 BIT FIELDS. THE 32 BITS ARE IN THE LOW ORDER PART OF EACH WORD. THE BIT IS ON IF AND ONLY IF THE CORRESPONDING PRT CELL HAS A PERMANENT ASSIGNMENT;	01699000 T	0098
	INTEGER PRTI, PRTIMAX;	01700000 T	0098
PRT(305) = PRTI		01701000 T	0098
PRT(306) = PRTIMAX		01702000 T	0098
	COMMENT PRTIMAX GIVES NEXT PRT CELL AVAILABLE FOR PERMANENT ASSIGN- MENT. PRTI GIVES NEXT PRT CELL POSSIBLY AVAILABLE FOR TEMPORARY ASSIGNMENT;	01703000 T	0098
	DEFINE ALPHASIZE = [12:6]; COMMENT ALPHASIZE IS THE DEFINE FOR THE BIT POSITION IN THE SECOND WORD OF INFO WHICH CONTAINS THE LENGTH OF ALPHA;	01704000 T	0098
	DEFINE EDOCINDEX = L.[36:3], L.[39:7]; COMMENT EDOCINDEX IS THE WORD PORTION OF L SPLIT INTO A ROW AND COLUMN INDEX FOR EDOC;	01705000 T	0098
	DEFINE CPLUS1 = 769; COMMENT SEE COMMENT AT CPLUS2 DEFINE;	01706000 T	0098
	DEFINE CPLUS2 = 770; COMMENT CPLUS1 AND CPLUS2 ARE EXPLICIT CONSTANTS USED IN THE GENERATION OF C-RELATIVE CODE;	01707000 T	0098
	PROCEDURE FLAG(ERRNUM); VALUE ERRNUM; INTEGER ERRNUM; FORWARD;	01708000 T	0098
PRT(307) = FLAG		01709000 T	0098
PRT(310) = B2D	ALPHA PROCEDURE B2D(B); VALUE B; REAL B; FORWARD;	01710000 T	0098
PRT(311) = TAKE	REAL PROCEDURE TAKE(W); VALUE W; INTEGER W; FORWARD;	01711000 T	0098
PRT(312) = MACROID	BOOLEAN MACROID;	01712000 T	0098
PRT(313) = FIXDEFINEINFO	REAL PROCEDURE FIXDEFINEINFO(T); VALUE T; REAL T; FORWARD;	01713000 T	0098
PRT(314) = ERR	PROCEDURE ERR (ERRNUM); VALUE ERRNUM; INTEGER ERRNUM; FORWARD;	01714000 T	0098
PRT(315) = GIT	INTEGER PROCEDURE GIT(L); VALUE L; REAL L; FORWARD;	01715000 T	0098
PRT(316) = CALLA	ARRAY CALLA[0:31, 0:255];	01716000 T	0098
	DEFINE CALL[CALL1] = CALLA[(GT3 * CALL1), LINKR, GT3, LINKC];	01717000 T	0098
	REAL CALLX, CALLINFO, NESTCTR, NESTCUR;	01717700 T	0098
PRT(317) = CALLX		01717800 T	0098
PRT(320) = CALLINFO		01717900 T	0098
PRT(321) = NESTCTR		01718000 T	0098
PRT(322) = NESTCUR		01719000 T	0098
		01720000 T	0098
		01721000 T	0100
		01722000 T	0100

<pre> BOOLEAN NESTOG; PRT(323) = NESTOG ARRAY NESTPRT[PRTBASE:PRTOP]; PRT(324) = NESTPRT ARRAY SORTPRT[0:PRTOP-PRTBASE]; PRT(325) = SORTPRT COMMENT "BLANKET" BLANKS OUT N+1 WORDS IN "THERE"; STREAM PROCEDURE BLANKET(N,THERE); VALUE N; PRT(326) = BLANKET BEGIN DI:=THERE; DS:=8 LIT " "; SI:=THERE; DS:=N WDS; END BLANKET; </pre>	<pre> 01723000 T 0100 01724000 T 0100 01725000 T 0103 01726000 T 0107 01737300 T 0107 01737350 T 0107 01737400 T 0107 01737450 T 0108 01737500 T 0110 </pre>
<pre> STREAM PROCEDURE CHANGESEQ(VAL,OLDSEQ); VALUE OLDSEQ; PRT(327) = CHANGESEQ BEGIN DI:=OLDSEQ; SI:=VAL; DS:=8 DEC END CHANGESEQ; </pre>	<pre> 01741200 T 0110 01741300 T 0110 </pre>
<pre> STREAM PROCEDURE SEQUENCEERROR(L); PRT(330) = SEQUENCEERROR BEGIN DI:=L; DS:=16 LIT"SEQUENCE ERROR "; END SEQUENCEERROR; </pre>	<pre> 01742100 T 0112 01742110 T 0112 </pre>
<pre> STREAM PROCEDURE GETVOID(VP,NCR,LCR,SEQ); VALUE NCR,LCR; PRT(331) = GETVOID BEGIN LABEL L,EXIT; LOCAL N; SI:=NCR; DI:=VP; DS:=8 LIT "0"; 2(34(IF SC=" " THEN SI:=SI+1 ELSE JUMP OUT 2 TO L)); GO TO EXIT; % NO VOID RANGE GIVEN, RETURN ZERO. L: IF SC="%" THEN GO TO EXIT; % STILL NO RANGE. IF SC=""" THEN BEGIN SI:=SI+1; DI:=LCR; DS:=1 LIT""; % STOPPER FOR SCAN NCR:=SI; % TEMP. STORAGE, SINCE NCR IS "LOCAL" TO GETVOID. 8(IF SC=""" THEN JUMP OUT ELSE BEGIN TALLY:=TALLY+1; SI:=SI+1 END); END ELSE BEGIN NCR:=SI; % TEMP. STORAGE, SINCE NCR IS "LOCAL" TO GETVOID. DI:=LCR; DS:=1 LIT " "; % STOPPER FOR SCAN 8(IF SC=" " THEN JUMP OUT ELSE BEGIN TALLY:=TALLY+1; SI:=SI+1 END); END; SI:=NCR; DI:=VP; DI:=DI+8; % RESTORE POINTERS. NI:=TALLY; DI:=DI-N; DS:=N CHR; EXIT; </pre>	<pre> 01756000 T 0114 01757000 T 0114 01758000 T 0115 01759000 T 0115 01761000 T 0115 01762000 T 0116 01763000 T 0119 01764000 T 0119 01764500 T 0119 01765000 T 0120 01766000 T 0121 01767000 T 0121 01768000 T 0122 01769000 T 0123 01770000 T 0124 01771000 T 0126 01772000 T 0126 01773000 T 0126 01774000 T 0126 01775000 T 0127 01776000 T 0128 01777000 T 0130 01780000 T 0130 01781000 T 0130 01782000 T 0132 </pre>

```

END OF GETVOID;

REAL VOIDCR,VOIDPLACE,VOIDTCR,VOIDTPLACE;
PRT(332) = VOIDCR
PRT(333) = VOIDPLACE
PRT(334) = VOIDTCR
PRT(335) = VOIDTPLACE
      FORMAT
      BUG(X24,4(A4,X2));

PRT(336) = BUG

      PROCEDURE DATIME;
PRT(337) = DATIME
      BEGIN
      INTEGER H,MIN,Q; ALPHA N1,N2;

STACK(F+2) = H
STACK(F+3) = MIN
STACK(F+4) = Q
STACK(F+5) = N1
STACK(F+6) = N2

      ALPHA STREAM PROCEDURE DATER(DATE); VALUE DATE;
      BEGIN
      DI:=LOC DATER; SI:=LOC DATE; SI:=SI+2;
      2(DS:=2 CHR; DS:=LIT"/"); DS:=2 CHR;
      END OF DATER;

      H:=TIME1 DIV 216000; MIN:=(TIME1 DIV 3600) MOD 60;
      N1:=DISK,MFID; N2:=DISK,FID;
      WRITE(LINE,
      <X22,"BURROUGHS B-5700 ESPOL COMPILER MARK ",
PRT(341) = *FORMAT DESCRIPTOR*
      "XVI,0.00"
      , " ",A6,"DAY, ",0," ",I2,"1",A2,X1,A3,
      ///X45,A1,A6,"/",A1,A6,/X45,15("=")//>,

      TIME(6),DATER(TIME(5)),12*REAL(Q:=H MOD 12=0)+Q,
PRT(342) = *LIST, LABEL, OR SEGMENT DESCRIPTOR*
      Q:=MIN MOD 10+(MIN DIV 10)*64,
      IF H<12 THEN "PM," ELSE "AM,"
      N1,[6:6],N1,N2,[6:6],N2);

PRT(343) = OUTPUT(W)
      NOHEADING:=FALSE;
      END OF DATIME;

```

```

01784000 T 0132

01785000 T 0132

01800000 T 0132
01802000 T 0132
START OF SEGMENT ***** 4
4 IS 8 LONG, NEXT SEG 3
01820000 T 0132

01821000 T 0132
01822000 T 0132
START OF SEGMENT ***** 5

01823000 T 0000
01824000 T 0000
01825000 T 0000
01826000 T 0000
01827000 T 0002

01828000 T 0003
01828500 T 0007
01829000 T 0013
01830000 T 0015

01831000 T 0015
01832000 T 0015
01832500 T 0015
6 IS 40 LONG, NEXT SEG 5
01833000 T 0016

01834000 T 0032
01835000 T 0036
01835500 T 0043

01836000 T 0053
01837000 T 0054
5 IS 58 LONG, NEXT SEG 3

```

COMMENT THIS SECTION CONTAINS ALL CODE PERTAINENT TO READING CARDS

02000000 T 0132

```

AND SCANNING THEM;
COMMENT OCTIZE REFORMATS ACCUM FOR OCTAL CONSTANTS;
BOOLEAN STREAM PROCEDURE OCTIZE(S,D,SKP,CNT); VALUE SKP,CNT;
PRT(344) = OCTIZE
BEGIN
SI:=S; SI:=SI+4; DI:=D; SKP(DS:=3 RESET); % RIGHT JUSTIFY,
CNT(IF SC<"0" THEN TALLY:=1 ELSE IF SC<"0" THEN TALLY:=1; SKIP 3 SB;
3(IF SB THEN DS:=SET ELSE DS:=RESET; SKIP SB));
SI:=D; IF SB THEN
BEGIN TALLY:=1; DI:=D; DS:=RESET END; % PREVENT FLAG BIT.
OCTIZE:=TALLY; % "1" = NON OCTAL CHARACTER OR FLAG BIT.
END OCTIZE;

```

```

COMMENT HEXIZE REFORMATS ACCUM FOR HEXADEcimal CONSTANTS;
BOOLEAN STREAM PROCEDURE HEXIZE(S,D,SKP,CNT); VALUE SKP,CNT;
PRT(345) = HEXIZE
BEGIN LOCAL T1,T2,TEMP2,TEMP1; LABEL AGIN;
COMMENT LOCAL VARIABLES ARE LOCATED IN REVERSE ORDER FROM THE
WAY THEY ARE DECLARED IN STREAM PROCEDURES;
DI:=LOC TEMP1; CNT(DS:=LIT"1"); % IN CASE A CHAR=A,B,C,D,OR F,
SI:=S; SI:=SI+3; DI:=LOC TEMP1; % WE MAY OVERFLOW INTO TEMP2.
CNT(IF SC<"0" THEN IF SC<"A" THEN IF SC<"F" THEN % WORK HARD.
BEGIN
T1:=SI; T2:=DI; DI:=T1; SI:=T2; % FLIP, MAN.
DS:=3 RESET; SI:=T1; DI:=T2; % FLIP BACK.
DS:=1 ADD; DI:=DI-1; SKIP 2 DB; DS:=1 SET; SKIP 3 DB;
GO AGIN;
END;
IF SC<"0" THEN TALLY:=1; DS:=CHR; % < 0 = NON-HEX CHARACTER.
AGIN:
);
SI:=LOC TEMP1; DI:=D; SKP(DS:=4 RESET); % RIGHT ADJUST CONSTANT.
CNT(SKIP 2 SB;
4(IF SB THEN DS:=SET ELSE DS:=RESET; SKIP SB)); % FINAL CONVERT.
SI:=D; IF SB THEN
BEGIN TALLY:=1; DI:=D; DS:=RESET END; % PREVENT FLAG BIT.
HEXIZE:=TALLY; % "1" IF PROGRAMMER GOOFED.
END HEXIZE;

```

```

COMMENT PUTSEQNO PUTS THE SEQUENCE NUMBER OF THE CARD-IMAGE
CURRENTLY BEING SCANNED INTO THE INFO TABLE IN CASE
IT IS NEEDED FOR FUTURE REFERENCE;
STREAM PROCEDURE PUTSEQNO(INFO,LCR); VALUE LCR;
PRT(346) = PUTSEQNO
BEGIN DI:=INFO; SI:=LCR; DS:=WDS; END PUTSEQNO;

```

```

COMMENT TURNONSTOPLIGHT TURNS THE LIGHT "RED" ON THE "CORNER".
I.E., THE PURPOSE OF THIS ROUTINE IS TO INSERT A PER-

```

```

02001000 T 0132
02001836 T 0132
02001838 T 0132

02001840 T 0132
02001842 T 0133
02001844 T 0135
02001846 T 0138
02001848 T 0140
02001850 T 0141
02001852 T 0142
02001854 T 0142

```

```

02001856 T 0143
02001858 T 0143

```

```

02001860 T 0143
02001862 T 0144
02001864 T 0144
02001866 T 0144
02001868 T 0146
02001870 T 0146
02001872 T 0150
02001874 T 0150
02001876 T 0151
02001878 T 0152
02001880 T 0153
02001882 T 0154
02001884 T 0154
02001886 T 0155
02001888 T 0155
02001890 T 0155
02001892 T 0157
02001894 T 0158
02001895 T 0160
02001896 T 0161
02001897 T 0162
02001898 T 0163

```

```

02002000 T 0164
02003000 T 0164
02004000 T 0164
02005000 T 0164

02006000 T 0164

```

```

02007000 T 0165
02008000 T 0165

```


CENT SIGN IN COLUMN 73 AS AN END OF CARD SENTINEL FOR THE SCANNER; STREAM PROCEDURE TURNONSTOPLIGHT(RED,CORNER); VALUE RED,CORNER; PRT(347) = TURNONSTOPLIGHT BEGIN DI:=CORNER; SI:=LOC CORNER; SI:=SI-1; DS:=CHR END;	02009000 T 0165 02010000 T 0165 02011000 T 0165 02012000 T 0165
COMMENT WRITNEW TRANSFERS THE CARD IMAGE TO THE NEWTAPE BUFFER AND REPORTS IF THE CARD MIGHT BE CONTROL CARD; BOOLEAN STREAM PROCEDURE WRITNEW(NEW,FCR); VALUE FCR; PRT(350) = WRITNEW BEGIN SI + FCR; IF SC ≠ "\$" THEN TALLY + 1; DI+NEW;DS+10 WDS; WRITNEW + TALLY END WRITNEW;	02014000 T 0166 02015000 T 0166 02016000 T 0166 02017000 T 0166 02018000 T 0168 02020000 T 0168
COMMENT MKABS CONVERTS A DESCRIPTOR TO AN ABSOLUTE ADDRESS; REAL STREAM PROCEDURE MKABS(A); PRT(351) = MKABS BEGIN DI + A; MKABS + DI END MKABS;	02021000 T 0169 02022000 T 0169 02023000 T 0169
REAL STREAM PROCEDURE CONV(ACCUM,SKP,N);VALUE SKP,N; PRT(352) = CONV BEGIN SI+ ACCUM; SI+SI+SKP;SI+SI+3;DI+LOC CONV;DS+N OCT END;	02041000 T 0171 02042000 T 0171 02043000 T 0172 02044000 T 0173
STREAM PROCEDURE MOVECHARACTERS(N,SORCE,SSKIP,DEST,DSKIP); PRT(353) = MOVECHARACTERS VALUE N,SSKIP,DSKIP; BEGIN SI+SORCE ; DI+DEST; SI+SI+SSKIP; DI+ DI+DSKIP ; DS + N CHR ; END ;	02045000 T 0174 02046000 T 0174 02047000 T 0174 02048000 T 0175 02049000 T 0175 02050000 T 0176 02051000 T 0177
COMMENT MOVECHARACTERS MOVES N CHARACTERS FROM THE SSKIP-TH CHAR IN "SORCE" TO THE DSKIP-TH CHAR IN "DEST", STREAM PROCEDURE MOVE(W)"WORDS FROM"(A)"TO"(B); VALUE W; PRT(354) = MOVE BEGIN SI + A; DI + B; DS + W WDS END;	02052000 T 0177 02053000 T 0177 02054000 T 0177 02055000 T 0177

```

        STREAM PROCEDURE RESIZE(FIEL);
PRT(355) = RESIZE
        BEGIN LOCAL T;
          SI=FIEL; DI=LOC T; DS=WDS;
          SI=T; DI=FIEL; DI=DI+1; SKIP 2 DB; DS=10 SET
        END;

```

```

02056000 T 0179
02057000 T 0179
02058000 T 0180
02059000 T 0180
02060000 T 0182

```

```

        COMMENT EQUAL COMPARES COUNT CHARACTERS LOCATED AT A AND B FOR
          EQUALITY. THIS ROUTINE IS USED IN THE LOOK-UP OF ALPHA
          QUANTITIES IN THE DIRECTORY;
        BOOLEAN STREAM PROCEDURE EQUAL(COUNT,A,B); VALUE COUNT;
PRT(356) = EQUAL
        BEGIN
          TALLY:=1; SI:=A; DI:=B;
          IF COUNT SC=DC THEN EQUAL:=TALLY
        END EQUAL;

```

```

02061000 T 0182
02061500 T 0182
02062000 T 0182
02062500 T 0182
02063000 T 0182
02063500 T 0183
02064000 T 0183
02064500 T 0184

```

```

        PROCEDURE READACARD; FORWARD;
PRT(357) = READACARD
        PROCEDURE DOLLARCARD; FORWARD;
PRT(360) = DOLLARCARD
        BOOLEAN PROCEDURE BOOLEXP; FORWARD;
PRT(361) = BOOLEXP
        PROCEDURE SCANNER;
PRT(362) = SCANNER

```

```

02065000 T 0185
02065500 T 0185
02065600 T 0185
02066000 T 0185

```

```

        BEGIN
        COMMENT "SCAN" IS THE STREAM PROCEDURE WHICH DOES THE ACTUAL SCANNING.
          IT IS DRIVEN BY A SMALL WORD MODE PROCEDURE CALLED "SCANNER",
          WHICH CHECKS FOR A QUANTITY BEING BROKEN ACROSS A CARD. "SCAN"
          IS CONTROLLED BY A VARIABLE CALLED "RESULT". "SCAN" ALSO
          INFORMS THE WORLD OF ITS ACTION BY MEANS OF THE SAME VARIABLE.
          HENCE THE VARIABLE "RESULT" IS PASSED BY BOTH NAME AND VALUE,
          THE MEANING OF "RESULT" AS INPUT IS:
          VALUE      MEANING
          =====
          0          INITIAL CODE = DEBLANK AND START TO FETCH THE
                     NEXT QUANTITY.
          1          CONTINUE BUILDING AN IDENTIFIER (INTERRUPTED BY
                     END-OF-CARD BREAK).
          2          LAST QUANTITY BUILT WAS SPECIAL CHARACTER, HENCE,
                     EXIT (INTERRUPTION BY END-OF-CARD BREAK IS NOT
                     IMPORTANT).
          3          CONTINUE BUILDING A NUMBER (INTERRUPTED BY END-OF-
                     CARD BREAK).
          4          LAST THING WAS AN ERROR (COUNT EXCEEDED 63), HENCE,
                     EXIT (INTERRUPTION BY END-OF-CARD BREAK NOT
                     IMPORTANT).
          5          GET NEXT CHARACTER AND EXIT.
          6          SCAN A COMMENT.
          7          DEBLANK ONLY.
        THE MEANING OF "RESULT" AS OUTPUT IS:

```

```

02066500 T 0185
02067000 T 0185
02067500 T 0185
02068000 T 0185
02068500 T 0185
02069000 T 0185
02069500 T 0185
02070000 T 0185
02070500 T 0185
02071000 T 0185
02071500 T 0185
02072000 T 0185
02072500 T 0185
02073000 T 0185
02073500 T 0185
02074000 T 0185
02074500 T 0185
02075000 T 0185
02075500 T 0185
02076000 T 0185
02076500 T 0185
02077000 T 0185
02077500 T 0185
02078000 T 0185
02078500 T 0185
02079000 T 0185

```

VALUE	MEANING
=====	=====
1	AN IDENTIFIER WAS BUILT.
2	A SPECIAL CHARACTER WAS OBTAINED.
3	A NUMBER (INTEGER) WAS BUILT.

"SCAN" PUTS ALL STUFF SCANNED (EXCEPT FOR COMMENTS AND DISCARDED BLANKS) INTO "ACCUM" (CALLED "ACCUMULATOR" FOR THE REST OF THIS DISCUSSION).

"COUNT" IS THE VARIABLE THAT GIVES THE NUMBER OF CHARACTERS "SCAN" HAS PUT INTO THE "ACCUMULATOR". SINCE "SCAN" NEEDS THE VALUE SO THAT IT CAN PUT MORE CHARACTERS INTO THE "ACCUMULATOR" AND NEEDS TO UPDATE "COUNT" FOR THE OUTSIDE WORLD, "COUNT" IS PASSED BY BOTH NAME AND VALUE. IT IS ALSO CONVENIENT TO HAVE (63-COUNT). THIS IS CALLED "COMCOUNT".

"NCR" (NEXT CHARACTER TO BE SCANNED) IS ALSO PASSED BY NAME AND VALUE SO THAT IT MAY BE UPDATED.

"ST1" AND "ST2" ARE TEMPORARY STORAGES WHICH ARE EXPLICITLY PASSED TO "SCAN" IN ORDER TO OBTAIN THE MOST USEFULL STACK ARRANGEMENT.

```

;
STREAM PROCEDURE SCAN(NCR,COUNTV,ACCUM,COMCOUNT,RESULT,RESULTV,

```

PRT(363) = SCAN

```

COUNT,ST2,NCRV,ST1);
VALUE COUNTV, COMCOUNT,RESULTV,ST2,NCRV,ST1;
BEGIN
LABEL DEBLANK,NUMBERS,IDBLDR,GNC,K,EXIT,FINIS,L,ERROR,
COMMENTS,COMMANTS;
DI:=RESULT; DI:=DI+7; SI:=NCRV;
COMMENT SETUP "DI" FOR A CHANGE IN "RESULT" AND "SI" FOR A LOOK AT
THE BUFFER;
CI:=CI+RESULTV; % SWITCH ON VALUE OF RESULT;
GO DEBLANK; % 0 IS INITIAL CODE.
GO IDBLDR; % 1 IS ID CODE.
GO FINIS; % 2 IS SPECIAL CHARACTER CODE.
GO NUMBERS; % 3 IS NUMBER CODE.
GO FINIS; % 4 IS ERROR CODE.
GO GNC; % 5 IS GET NEXT CHARACTER CODE.
GO COMMANTS; % 6 IS COMMENT CODE.
% 7 IS DEBLANK ONLY CODE.
IF SC=" " THEN
K: BEGIN SI:=SI+1; IF SC=" " THEN GO K END;
GO FINIS;
DEBLANK: IF SC=" " THEN
L: BEGIN SI:=SI+1; IF SC=" " THEN GO L END;
COMMENT IF WE ARRIVE HERE WE HAVE A NON-BLANK CHARACTER;
NCRV:=SI;
IF SC >= "0" THEN GO NUMBERS;
IF SC=ALPHA THEN GO IDBLDR;
COMMENT IF WE ARRIVE HERE WE HAVE A SPECIAL CHARACTER (OR GNC);
GNC: DS:=LIT"2"; TALLY:=1; SI:=SI+1; GO EXIT;
COMMANTS: IF SC=";" THEN
BEGIN
COMMENTS:

```

02079500	T	0185
02080000	T	0185
02080500	T	0185
02081000	T	0185
02081500	T	0185
02082000	T	0185
02082500	T	0185
02083000	T	0185
02083500	T	0185
02084000	T	0185
02084500	T	0185
02085000	T	0185
02085500	T	0185
02086000	T	0185
02086500	T	0185
02087000	T	0185
02087500	T	0185
02088000	T	0185
02088500	T	0185
02089000	T	0185
02089500	T	0185
START OF SEGMENT ***** 7		
02090000	T	0000
02090500	T	0000
02091000	T	0000
02091500	T	0000
02092000	T	0000
02092500	T	0000
02093000	T	0000
02093500	T	0000
02094000	T	0000
02094500	T	0001
02095000	T	0001
02095500	T	0001
02096000	T	0002
02096500	T	0002
02097000	T	0002
02097500	T	0002
02098000	T	0003
02098500	T	0003
02099000	T	0003
02099500	T	0005
02100000	T	0005
02100500	T	0005
02101000	T	0006
02101500	T	0008
02102000	T	0008
02102500	T	0008
02103000	T	0009
02103500	T	0009
02104000	T	0009
02104500	T	0009
02105000	T	0011
02105500	T	0011
02106000	T	0012
02106500	T	0012

```

        SII=SI+1;
        IF SC > "%" THEN GU COMMENTS;
        IF SC < ";" THEN GU COMMENTS;
COMMENT  CHARACTERS BETWEEN % AND SEMICOLON ARE HANDLED BY WORD=
        MODE PART OF COMMENT ROUTINE;
        END;
        GO FINIS;
IDBLDR:  TALLY:=63; DS=LIT "1";
        COMCOUNT(TALLY:=TALLY+1;
        IF SC=ALPHA THEN SII=SI+1 ELSE JUMP OUT TO EXIT);
        TALLY:=TALLY+1;
        IF SC=ALPHA THEN
            BEGIN
ERROR:    DI:=DI-1; DS=LIT "4"; GO EXIT;
            END
        ELSE GO EXIT;
COMMENT  IF WE ARRIVE AT ERROR WE HAVE MORE THAN 63 CHARACTERS
        IN AN IDENTIFIER OR NUMBER;
NUMBERS: TALLY:=63; DS=LIT "3";
PRT(364) = *LIST, LABEL, OR SEGMENT DESCRIPTOR*
        COMCOUNT(TALLY:=TALLY+1;
        IF SC < "0" THEN JUMP OUT TO EXIT; SII=SI+1);
        GO ERROR;
EXIT:    ST1:=TALLY; % "ST1" CONTAINS NUMBER OF CHARACTERS WE ARE
        % GOING TO MOVE INTO THE "ACCUMULATOR".
        TALLY:=TALLY+COUNTV; ST2:=TALLY;
        DI:=COUNT; SII=LOC ST2; DS=WDS;
COMMENT  THIS CODE UPDATED "COUNT";
        DI:=ACCUM; SII=SI-3; DS=3 CHR;
COMMENT  THIS CODE PLACES "COUNT" IN "ACCUM" AS WELL;
        DI:=DI+COUNTV; % POSITION "DI" PAST CHARACTERS ALREADY
        % IN THE "ACCUMULATOR", IF ANY.
        SII=NCRV; DS=ST1 CHR;
COMMENT  MOVE CHARACTERS INTO "ACCUM";
FINIS:   DI:=NCR; ST1=SI; SII=LOC ST1; DS=WDS;
PRT(365) = *LIST, LABEL, OR SEGMENT DESCRIPTOR*
COMMENT  RESET "NCR" TO LOCATION OF NEXT CHARACTER TO BE SCANNED;
        END OF SCAN;

```

```

02107000 T 0013
02107500 T 0013
02108000 T 0014
02108500 T 0014
02109000 T 0014
02109500 T 0014
02110000 T 0014
02110500 T 0015
02111000 T 0015
02111500 T 0015
02112000 T 0017
02112500 T 0018
02113000 T 0019
02113500 T 0019
02114000 T 0019
02114500 T 0020
02115000 T 0021
02115500 T 0021
02116000 T 0021
02116500 T 0021
02117000 T 0021
02117500 T 0021

02118000 T 0022
02118500 T 0024
02119000 T 0025
02119500 T 0025
02120000 T 0025
02120500 T 0026
02121000 T 0026
02121500 T 0027
02122000 T 0027
02122500 T 0027
02123000 T 0028
02123500 T 0028
02124000 T 0029
02124500 T 0029
02125000 T 0029
02125500 T 0029
02126000 T 0029

02126500 T 0031
02127000 T 0031

```

```

        LABEL L;%
L:       SCAN(NCR,COUNT,ACCUM[1],63-COUNT,RESULT,
        RESULT,COUNT,0,NCR,0);
        IF NCR=LCR THEN
            BEGIN
                READACARD;
                GO TO L; % GO DIRECTLY TO L, DO NOT PASS GO,
                % DO NOT COLLECT $200.
            END;

```

```

02127500 T 0031
02128000 T 0031
02128500 T 0032
02129000 T 0034
02129500 T 0036
02130000 T 0037
02130500 T 0037
02131000 T 0038
02131500 T 0038
02132000 T 0038
02132500 T 0038

```

END SCANNER;

02137000 T 0038
7 IS 39 LONG, NEXT SEG 3

```
DEFINE WRITELINE = IF SINGLTOG THEN WRITE(LINE,15,LIN[*])
                    ELSE WRITE(LINE[DBL],15,LIN[*])#;
PRINTCARD = BEGIN
    EDITLINE(LIN,FCR,L DIV 4,L.[46:2],MEDIUM,OMITTING);
    IF NOHEADING THEN DATIME; WRITELINE;
    END #;
STREAM PROCEDURE EDITLINE(LINE,NCR,R,L,SYMBOL,OMIT);
PRT(366) = EDITLINE
                    VALUE NCR,R,L,SYMBOL,OMIT;
BEGIN
    DI := LINE; DS := 16 LIT " ";
    SI := NCR; DS := 9 WDS;
    DS := 8 LIT " ";
    DS := WDS; % SEQUENCE NUMBER.
    DS:=LIT" "; SI:=LOC SYMBOL; SI:=SI+6;
    DS:=2 CHR; DS:=LIT" ";
    SI+LOC R; DS+4 DEC; DS+LIT "1";
    SI+LOC L; DS+1 DEC;
    DS+6 LIT " ";
    OMIT(DI:=DI-12; DS:=8 LIT"   OMIT");
END EDITLINE;
```

02181000 T 0185
02181250 T 0185
02182500 T 0185
02182750 T 0185
02183000 T 0185
02183250 T 0185
02183500 T 0185

02183750 T 0185
02184000 T 0185
02184250 T 0186
02184500 T 0188
02184750 T 0189
02185000 T 0190
02185250 T 0190
02185500 T 0191
02185750 T 0192
02186000 T 0193
02186250 T 0193
02186750 T 0194
02187000 T 0197

```
COMMENT COMPARE COMPARES SEQUENCE NUMBERS OF TAPE AND CARD. IF
    TAPE IS SMALLER THEN RESULT = 0 ELSE IF CARD IS SMALLER
    RESULT = 1 ELSE RESULT = 2;
REAL STREAM PROCEDURE COMPARE(TAPE,CARD); VALUE TAPE,CARD;
PRT(367) = COMPARE
    BEGIN
    SI := TAPE; DI := CARD;
    IF 8 SC 2 DC THEN
        BEGIN
            SI := SI-8; DI := DI-8; TALLY := 1;
            IF 8 SC = DC THEN TALLY := 2
        END;
    COMPARE := TALLY
    END COMPARE;
```

02187250 T 0197
02187500 T 0197
02187750 T 0197
02188000 T 0197

02188250 T 0197
02188500 T 0198
02188750 T 0198
02189000 T 0199
02189250 T 0199
02189500 T 0200
02189750 T 0200
02190000 T 0201
02190250 T 0201

```
PROCEDURE OUTPUTSOURCE;
PRT(370) = OUTPUTSOURCE
    BEGIN
    LABEL LCARD,LTAPE,AWAY;

    SWITCH SW:=LCARD,LCARD,LTAPE,AWAY,LCARD,LTAPE;
    IF SEQTOG THEN % RESEQUENCING,
        BEGIN
```

02190500 T 0202

02190750 T 0202
02191000 T 0202
START OF SEGMENT ***** 8
02191250 T 0000
02191500 T 0006
02191750 T 0007

```

        IF TOTALNO = -10 OR NEWBASE THEN
            BEGIN
                NEWBASE := FALSE; GTI1 := TOTALNO := BASENUM
            END
        ELSE GTI1 := TOTALNO := TOTALNO + ADDVALUE;
        CHANGESEQ(GTI1,LCR);
    END;
    IF NEWTOG THEN
        IF WRITNEW(LIN,FCR) THEN WRITE(NEWTAPE,10,LIN[*]);
        IF OMITTING THEN IF NOT LISTATOG THEN GO AWAY;
        GO SW(LASTUSED);
    LCARD:
        IF LISTER OR LISTPTOG THEN PRINTCARD;
        GO AWAY;
    LTAPE:
        IF LISTER THEN PRINTCARD;
        GO AWAY;
    % GO AWAY;
    AWAY:
        END OUTPUTSOURCE;

```

```

02192000 T 0008
02192250 T 0009
02192500 T 0010
02192750 T 0011
02193000 T 0012
02193250 T 0014
02193500 T 0015
02193750 T 0015
02194000 T 0016
02194250 T 0023
02194500 T 0025
02194750 T 0027
02195000 T 0028
02195250 T 0045
02195500 T 0045
02195750 T 0046
02196000 T 0062
02196250 T 0062
02196500 T 0062
8 IS 63 LONG, NEXT SEG 3

```

```

PROCEDURE READACARD;
COMMENT READACARD READS CARDS FROM EITHER THE CARD READER OR THE
        TAPE MERGING AS REQUESTED AND CREATING A NEW TAPE AND
        LISTING IF REQUESTED. READACARD ALSO INSERTS A PERCENT
        SIGN AS AN END OF CARD SENTINEL IN COLUMN 73 AND SETS
        FCR, NCR, LCR, TLCR, AND CLCR;
    BEGIN
        PROCEDURE READTAPE;

```

PRT(371) = READTAPE

```

        BEGIN
        LABEL ENDREADTAPE, EOFT;

```

```

        READ (TAPE, 10, TBUF[*])(EOFT);

```

PRT(372) = EOFT

PRT(373) = GO TO SOLVER

```

        LCR := MKABS(TBUF[9]);
        GO TO ENDREADTAPE;
    EOFT:
        DEFINEARRAY[25] := "ND;END."& "E"[1:43:5];
        DEFINEARRAY[34] := "9999" & "9999"[1:25:23];
        TLCR := MKABS(DEFINEARRAY[34]);
        PUTSEQNO (DEFINEARRAY[33],TLCR=8);
        TURNONSTOPLIGHT("X", TCR=8);
    ENDREADTAPE;
        END READTAPE;

```

```

02196750 T 0202
02197000 T 0202
02197250 T 0202
02197500 T 0202
02197750 T 0202
02198000 T 0202
02198250 T 0202
02198500 T 0202
START OF SEGMENT ***** 9

```

```

02201500 T 0000
02201510 C 0000
START OF SEGMENT ***** 10
02201750 P 0000

```

```

02202000 P 0005
02202010 C 0007
02202020 C 0007
02202030 C 0008
02202040 C 0010
02202050 C 0012
02202060 C 0014
02202070 C 0016
02202080 C 0017
02202250 T 0018
10 IS 24 LONG, NEXT SEG 9

```

```

        PROCEDURE SEQCOMPARE(TLCR,CLCR, LIB); VALUE LIB; BOOLEAN LIB;
PRT(374) = SEQCOMPARE

```

02202500 T 0000

```

REAL TLCR, CLCR ;

BEGIN
MEDIUM:="C "; % CARD READER.
IF GT1:=COMPARE(TLCR,CLCR)=0 THEN % TAPE HAS LOW SEQUENCE NUMB
    BEGIN
        LCR:=TLCR; LASTUSED:=3;
        MEDIUM:="T "; % TAPE INPUT.
    END
ELSE BEGIN
    IF GT1 ≠ 1 THEN % TAPE AND CARD HAVE SAME SEQ
        BEGIN
            MEDIUM:="P "; % CARD PATCHES TAPE.
            READTAPE;
        END;
        LCR:=CLCR;
        LASTUSED:=2;
    END;
END OF SEQCOMPARE;

```

```

LABEL CARDONLY, CARDLAST, TAPELAST, EXIT, FIRSTTIME,
EOF, USETHESWITCH,
COMPAR, TESTVOID, XIT;
SWITCH USESWITCH:=CARDONLY,CARDLAST,TAPELAST,FIRSTTIME;
IF ERRORCOUNT>ERRMAX THEN ERN(611); % ERR LIMIT EXCEEDED - STOP.
USETHESWITCH:
    DOLLAR2TOG:=FALSE;
    GO TO USESWITCH[LASTUSED];
    MOVE(1,INFO[LASTUSED,LINKR,LASTUSED,LINKC],
        DEFINEARRAY[DEFINEINDEX=2]);
    LASTUSED := LASTUSED + 1;
    NCR := LCR-1;
    GO TO XIT;
FIRSTTIME:
    READ(CARD,10,CBUFF[*]);
    FCR:=NCR:=(LCR:=MKABS(CBUFF[9]))-9;
    MEDIUM:="C ";
    IF EXAMIN(FCR)≠"$" AND LISTER THEN PRINTCARD;
    PUTSEQNO(INFO[LASTSEQROW,LASTSEQUENCE],LCR);
    TURNONSTOPLIGHT("%",LCR);
    GO XIT;
COMMENT WE HAVE JUST INITIALIZED CARD INPUT;
CARDONLY:
    READ(CARD,10,CBUFF[*]);
    LCR := MKABS(CBUFF[9]); GO EXIT;
CARDLAST:
    READ(CARD,10,CBUFF[*])[EOF];
    CLCR := MKABS(CBUFF[9]);
    GO COMPAR;
EOF:
    DEFINEARRAY[25]:="ND;END."&"E"[1:43:5];
    DEFINEARRAY[34]:="9999"&"9999"[1:25:23];
    CLCR:=MKABS(DEFINEARRAY[34]);
    PUTSEQNO(DEFINEARRAY[33],CLCR=8);

```

PRT(375) = EOF

```

02202750 T 0000
02203000 T 0000
02203250 T 0000
02203500 T 0000
02203750 T 0003
02204000 T 0003
02204250 T 0005
02204500 T 0006
02204750 T 0006
02205000 T 0009
02205250 T 0009
02205500 T 0010
02208500 T 0011
02208750 T 0011
02209000 T 0011
02209250 T 0012
02209500 T 0013
02209750 T 0013

02210000 T 0015
02210250 T 0015
02210500 T 0015
02210750 T 0015
02211500 T 0020
02211750 T 0023
02211800 T 0023
02212000 T 0023
02212250 T 0025
02212500 T 0028
02212750 T 0029
02213000 T 0031
02213250 T 0032
02213500 T 0032
02213750 T 0033
02214000 T 0037
02214100 T 0040
02214200 T 0041
02214250 T 0061
02214500 T 0063
02214750 T 0064
02215000 T 0064
02215250 T 0064
02215500 T 0065
02215750 T 0069
02216000 T 0071
02216250 T 0072

02216500 T 0077
02216750 T 0079
02217000 T 0079
02217250 T 0080
02217500 T 0082
02217750 T 0084
02218000 T 0086

```



```

        TURNONSTOPLIGHT("%",CLCR=8);
%
        GO COMPARE;
COMMENT THIS RELEASES THE PREVIOUS CARD FROM CARD READER AND
        SETS UP CLCR;
TAPELAST;
        READTAPE;
COMMENT THIS RELEASES THE PREVIOUS CARD FROM TAPE AND SETS UP TLCH;
COMPARE;
        SEQCOMPARE(TLCR,CLCR,FALSE);
EXIT;
        NCR := FCR := LCR - 9;
COMMENT SETS UP NCR AND FCR;
        IF EXAMIN(FCR)!="S" THEN % CARDS DON'T COUNT.
                IF COMPARE(MKABS(INFO[LASTSEQROW, LASTSEQUENCE]),LCR)=1 THEN
                        BEGIN
                                FLAG(610); % SEQUENCE ERROR,
                                SEQUENCEERROR(LIN);
                        END;
                CARDNUMBER:=CONV(INFO[LASTSEQROW, LASTSEQUENCE-1],5,8);
                IF LASTUSED=3 THEN
                        BEGIN
                                IF VOIDTAPE THEN GO USETHESWITCH;
                                IF VOIDTCR#0 THEN
                                        IF COMPARE(LCR,VOIDTCR)=0 THEN GO USETHESWITCH;
                        END;
                IF EXAMIN(FCR)="S" THEN
                        BEGIN
                                IF LISTPTOG OR PRINTDOLLARTOG THEN PRINTCARD;
                                NCR:=NCR+32768; DOLLARCARD;
COMMENT DONT FORGET THAT NCR IS NOT WORD MODE, BUT CHAR. MODE POINTER;
                                GO USETHESWITCH;
                        END;
                IF EXAMIN(FCR)=" " THEN
                        IF DOLLAR2TOG:=EXAMIN(FCR+32768)!="S" THEN
                                BEGIN
                                        OUTPUTSOURCE;
                                        NCR:=NCR+65536; % SCAN PAST " S" (CHARACTER MODE).
                                        DOLLARCARD;
                                END;
                IF VOIDING THEN GO USETHESWITCH;
                IF VOIDCR#0 THEN
                        IF COMPARE(LCR,VOIDCR)>0 THEN VOIDCR:=VOIDPLACE:=0
                                ELSE GO USETHESWITCH;
                IF VOIDTAPE THEN GO TESTVOID;
                IF VOIDTCR#0 THEN
                        IF COMPARE(LCR,VOIDTCR)>0 THEN VOIDTCR:=VOIDTPLACE:=0 ELSE
TESTVOID: IF LASTUSED=3 THEN GO USETHESWITCH;
                CARDCOUNT:=CARDCOUNT+1;
                IF DOLLAR2TOG THEN GO USETHESWITCH;
                PUTSEQNO(INFO[LASTSEQROW, LASTSEQUENCE],LCR);
                OUTPUTSOURCE;
                IF OMITTING THEN GO USETHESWITCH;
%
        TURNONSTOPLIGHT("%",LCR);
XIT:
        END READACARD;

```

```

02218250 T 0088
02218400 T 0089
02218500 T 0089
02218750 T 0093
02219000 T 0093
02219250 T 0093
02219500 T 0093
02219750 T 0093
02224250 T 0093
02224500 T 0094
02225000 T 0095
02225250 T 0096
02225500 T 0097
02225750 T 0097
02226000 T 0099
02226250 T 0104
02226500 T 0104
02226750 T 0105
02227000 T 0106
02228000 T 0106
02228050 T 0110
02228075 T 0110
02228100 T 0111
02228125 T 0112
02228150 T 0113
02228175 T 0116
02228250 T 0116
02228500 T 0118
02228750 T 0118
02229000 T 0136
02229250 T 0138
02229500 T 0138
02229750 T 0140
02230000 T 0140
02230100 T 0141
02230250 T 0145
02230500 T 0145
02230750 T 0146
02231000 T 0147
02231250 T 0147
02231500 T 0147
02231750 T 0149
02232000 T 0150
02232250 T 0153
02232500 T 0154
02233000 T 0155
02233500 T 0156
02234000 T 0160
02234500 T 0165
02234600 T 0166
02234750 T 0167
02235000 T 0169
02235250 T 0170
02235500 T 0171
02235750 T 0171
02237750 T 0172
02238000 T 0173

```

REAL PROCEDURE CONVERT;
PRT(376) = CONVERT

STACK(F+3) = T
STACK(F+4) = N

```
BEGIN REAL T; INTEGER N;

TLO:=0; THI:=
  T:= CONV(ACCUM[1],TCOUNT,N+(COUNT-TCOUNT)MOD 8);
  FOR N:= TCOUNT+N STEP 8 UNTIL COUNT= 1 DO
  IF DPTOG THEN
  BEGIN
  DOUBLE(THI,TLO,100000000.0,0,x,CONV(ACCUM[1],N,8),0,+,+,
    THI,TLO);
  T:=THI;
  END ELSE
  T:= T*100000000+ CONV(ACCUM[1],N,8);
  CONVERT:=T;
END;
```

02248000 T 0202

02249000 T 0202

START OF SEGMENT ***** 11

02250000 T 0000

02251000 T 0000

02252000 T 0005

02253000 T 0010

02254000 T 0010

02255000 T 0010

02256000 T 0015

02257000 T 0016

02258000 T 0016

02259000 T 0016

02260000 T 0024

02261000 T 0024

11 IS 28 LONG, NEXT SEG 3

REAL STREAM PROCEDURE FETCH(F); VALUE F;
PRT(377) = FETCH

```
BEGIN SI:=F; SI:=SI*8; DI:=LOC FETCH; DS:=WDS END FETCH;
```

02262000 T 0202

02263000 T 0202

PROCEDURE DUMPINFO;
PRT(400) = DUMPINFO

```
BEGIN
  ARRAY A[0:14]; INTEGER JEDEN=DWA;
```

STACK(F+2) = A
STACK(F+3) = JEDEN
STACK(F+4) = DWA

STREAM PROCEDURE OCTALWORDS(S,D,N); VALUE N;
PRT(401) = OCTALWORDS

```
BEGIN
  SI:=S; DI:=D;
  N(2(8(DS:=3 RESET; 3(IF SB THEN DS:=1 SET ELSE
    DS:=1 RESET; SKIP 1 SB))) DS:=1 LIT " ");DS:=2 LIT " ");
  END OF OCTALWORDS;
```

02264000 T 0205

02264050 T 0205

02264100 T 0205

START OF SEGMENT ***** 12

02264400 T 0001

02264450 T 0001

02264500 T 0003

02264550 T 0003

02264600 T 0006

02264650 T 0009

STREAM PROCEDURE ALPHAWORDS(S,D,N); VALUE N;
PRT(402) = ALPHAWORDS
BEGIN

02264700 T 0009

02264750 T 0009

```

SI:=S; DI:=D;
N(2(4(DS:=1 LIT" "; DS:=1 CHR); DS:=1 LIT" "); DS:=2 LIT" ");
END OF ALPHAWORDS;

```

```

02264800 T 0010
02264850 T 0010
02264900 T 0014

```

```

IF NOHEADING THEN DATIME;WRITE(LINE[DBL],</"ELBAT">);
PRT(403) = *FORMAT DESCRIPTOR*

```

```

02264950 T 0014

```

```

FOR JEDEN:=0 STEP 6 UNTIL 71 DO
  BEGIN
    BLANKET(14,A); OCTALWORDS(ELBAT[JEDEN],A,6);
    WRITE(LINE[DBL],15,A[*]);
  END;
  BLANKET(14,A); OCTALWORDS(ELBAT[72],A,4);
  WRITE(LINE[DBL],15,A[*]);
  FOR JEDEN:=0 STEP 1 UNTIL NEXTINFO DIV 256 DO
    BEGIN
      WRITE(LINE[DBL],</"INFO["",12,"",*]">,JEDEN);

```

```

PRT(404) = *FORMAT DESCRIPTOR*

```

```

PRT(405) = *LIST, LABEL, OR SEGMENT DESCRIPTOR*

```

```

FOR DWA:=0 STEP 6 UNTIL 251 DO
  BEGIN
    BLANKET(14,A); ALPHAWORDS(INFO[JEDEN,DWA],A,6);
    WRITE(LINE,15,A[*]);
    BLANKET(14,A); OCTALWORDS(INFO[JEDEN,DWA],A,6);
    WRITE(LINE[DBL],15,A[*]);
  END;
  BLANKET(14,A); ALPHAWORDS(INFO[JEDEN,252],A,4);
  WRITE(LINE,15,A[*]);
  BLANKET(14,A); OCTALWORDS(INFO[JEDEN,252],A,4);
  WRITE(LINE[DBL],15,A[*]);
END;
END OF DUMPINFO;

```

```

13 IS      6 LONG, NEXT SEG      12

```

```

02265000 T 0019
02265050 T 0020
02265100 T 0020
02265150 T 0023
02265200 T 0027
02265250 T 0029
02265300 T 0032
02265350 T 0036
02265400 T 0041
02265450 T 0041

```

```

14 IS      8 LONG, NEXT SEG      12

```

```

02265500 T 0048
02265550 T 0049
02265600 T 0049
02265650 T 0052
02265700 T 0057
02265750 T 0060
02265800 T 0065
02265850 T 0067
02265900 T 0071
02265950 T 0075
02266000 T 0079
02266050 T 0083
02266100 T 0083

```

```

12 IS     88 LONG, NEXT SEG      3

```

```

DEFINE SKAN = BEGIN
  COUNT:=RESULT:=ACCUM[1]:=0;
  SCANNER;
  Q:=ACCUM[1];
END #;
COMMENT DOLLARCARD HANDLES THE COMPILER CONTROL CARDS.
ALL COMPILER- AND USER-DEFINED OPTIONS ARE KEPT
IN THE ARRAY "OPTIONS".
EACH OPTION HAS A TWO-WORD ENTRY:

```

WORD	CONTAINS
----	-----
1	ENTRY FROM ACCUM[1]; 00XZZZZ, WHERE X IS THE SIZE OF THE ID AND ZZZZZ IS THE FIRST FIVE CHARS OF THE ID.
2	PUSH-DOWN, 47-BIT STACK CONTAINING THE

```

02277000 T 0205
02278000 T 0205
02279000 T 0205
02280000 T 0205
02281000 T 0205
02282000 T 0205
02283000 T 0205
02284000 T 0205
02285000 T 0205
02286000 T 0205
02287000 T 0205
02288000 T 0205
02289000 T 0205
02290000 T 0205
02291000 T 0205
02292000 T 0205

```

HISTORY OF THE SETTINGS OF THIS OPTION.

IN "FINDOPTION", ALL COMPILER-DEFINED OPTIONS ARE USUALLY LOCATED BASED UPON A UNIQUE NUMBER ASSIGNED TO EACH. FOR ALL USER-DEFINED OPTIONS, A SEQUENTIAL TABLE SEARCH IS INITIATED USING "USEROPINX" AS THE INITIAL INDEX INTO THE "OPTIONS" ARRAY. IF THE NUMBER OF COMPILER-DEFINED OPTIONS IS CHANGED, THEN "USEROPINX" MUST BE ACCORDINGLY CHANGED. THE NUMBER OF USER DEFINED OPTIONS ALLOWED CAN BE CHANGED BY CHANGING THE DEFINE "OPARSIZE". THE VARIABLE "OPTIONWORD" CONTAINS THE CURRENT TRUE OR FALSE SETTING OF ALL OF THE COMPILER-DEFINED OPTIONS, ONE BIT PER OPTION.

```

      ]
      BOOLEAN PROCEDURE FINDOPTION(BIT); VALUE BIT; INTEGER BIT;
PRT(406) = FINDOPTION
      BEGIN
        LABEL FOUND;
      REAL ID;
      STACK(F+3) = ID
      OPINX:=2*BIT-4;
      WHILE ID:=OPTIONS[OPINX:=OPINX+2]#0 DO
        IF Q=ID THEN GO FOUND;
      OPTIONS[OPINX]:=Q; % NEW USER-DEFINED OPTION.
FOUND:
      IF OPINX +1>OPARSIZE THEN FLAG(602) ELSE % TOO MANY USER OPTIUNS
      FINDOPTION:=BOOLEAN(OPTIONS[OPINX+1]);
      END FINDOPTION;

```

```

      PROCEDURE DOLLARCARD;
      BEGIN
        STREAM PROCEDURE RESTORESEQNUM(LCR,INFO); VALUE LCR;
PRT(407) = RESTORESEQNUM
      BEGIN
        DI:=LCR; SI:=INFO; DS:=WDS;
      END;

```

```

      PROCEDURE SWITCHIT(XBIT); VALUE XBIT; INTEGER XBIT;
PRT(410) = SWITCHIT
      BEGIN
        BOOLEAN B,T;
      INTEGER SAVEINX;
      LABEL XMODE0,XMODE1,XMODE2,XMODE3,XMODE4,ALONG;
      SWITCH SW:=XMODE0,XMODE1,XMODE2,XMODE3,XMODE4;

```

```

02293000 T 0205
02294000 T 0205
02295000 T 0205
02296000 T 0205
02297000 T 0205
02298000 T 0205
02299000 T 0205
02300000 T 0205
02301000 T 0205
02302000 T 0205
02303000 T 0205
02304000 T 0205
02305000 T 0205
02306000 T 0205
02307000 T 0205

02308000 T 0205
02309000 T 0205
START OF SEGMENT ***** 15
02310000 T 0000

02311000 T 0000
02312000 T 0001
02313000 T 0005
02314000 T 0006
02315000 P 0008
02316000 P 0008
02317000 T 0010
02318000 T 0012
15 IS 15 LONG, NEXT SEG 3

02319000 T 0205
02320000 T 0205
02320200 T 0205
START OF SEGMENT ***** 16
02320400 T 0000
02320600 T 0000
02320800 T 0000

02321000 T 0001

02322000 T 0001
02323000 T 0001
START OF SEGMENT ***** 17
02324000 T 0000

02325000 T 0000
02326000 T 0000

```

```

        SETTING:=FINDOPTION(XBIT); SKAN;
        GO SW[XMODE+1];
XMODE0:  % FIRST OPTION ON CARD, BUT NOT SET, RESET, OR POP.
        OPTIONWORD:=BOOLEAN(0);
        FOR SAVEINX:=1 STEP 2 UNTIL OPARSIZE DO OPTIONS[SAVEINX]:=0;
        XMODE:=LASTUSED:=1; % CARD INPUT ONLY.
XMODE1:  % NOT FIRST OPTION AND NOT BEING SET, RESET, OR POPPED.
        OPTIONS[OPINX+1]:=REAL(TRUE);
        IF XBIT<USEROPINX THEN OPTIONWORD:=OPTIONWORD & TRUE[XBIT:1];
PRT(411) = DYNAMIC DIALS
        GO ALONG;
XMODE2:  % RESET.
        OPTIONS[OPINX+1]:=REAL(FALSE & SETTING[1:46]);
        IF XBIT<USEROPINX THEN OPTIONWORD:=OPTIONWORD & FALSE[XBIT:1];
        GO ALONG;
XMODE3:  % SET.
        SAVEINX:=OPINX; % REMEMBER OPTION WE ARE SETTING.
        B:=IF Q="1=0000" THEN BOOLEXP ELSE TRUE;
        OPTIONS[SAVEINX+1]:=REAL(B & SETTING[1:46]);
        IF XBIT<USEROPINX THEN OPTIONWORD:=OPTIONWORD & B [XBIT:1];
        GO ALONG;
XMODE4:  % POP.
        OPTIONS[OPINX+1]:=REAL(B:=SETTING[1:46]);
        IF XBIT<USEROPINX THEN OPTIONWORD:=OPTIONWORD & B [XBIT:1];
ALONG:
        END SWITCHIT;

```

```

02327000 T 0005
02328000 T 0011
02329000 T 0013
02330000 T 0014
02331000 T 0014
02332000 T 0019
02333000 T 0020
02334000 T 0021
02335000 T 0022

```

```

02336000 T 0026
02337000 T 0027
02338000 T 0028
02339000 T 0030
02340000 T 0034
02341000 T 0035
02342000 T 0036
02343000 T 0036
02352000 T 0039
02353000 T 0042
02354000 T 0046
02355000 T 0048
02356000 T 0048
02357000 T 0051
02358000 T 0055
02359000 T 0056

```

17 IS 59 LONG, NEXT SEG 16

```

        LABEL EXIT, AGAIN, SKANAGAIN, LENGTH1, LENGTH2, LENGTH3, LENGTH4,
        LENGTH5, LENGTH6, LENGTH7, LENGTH8, LENGTH9,
        WHATISIT,
        CARDOPTION, MERGEOPTION;
        SWITCH OPTIONLENGTH:=LENGTH1, WHATISIT, LENGTH3, LENGTH4, LENGTH5,
        LENGTH6, LENGTH7, WHATISIT, LENGTH9, WHATISIT;
        INTEGER SRESULT, SCOUNT;
STACK(F+2) = SRESULT
STACK(F+3) = SCOUNT
        ALPHA VOIDRANGE;
STACK(F+4) = VOIDRANGE
        DOLLARTOG:=TRUE;
        MOVE(10, ACCUM[0], DEFINEARRAY[0]); % SAVE INFORMATION FOR
        SCOUNT:=COUNT; SRESULT:=RESULT; % "TABLE" TO RESUME SCAN.
        XMODE:=0;
        PUTSEQNO(INFO[LASTSEQRON, LASTSEQUENCE], LCR);
        TURNONSTOPLIGHT("%", LCR);
SKANAGAIN:
        SKAN;
AGAIN:
        GO OPTIONLENGTH[MIN(COUNT, 10)];
LENGTH1:
        IF Q = "1=0000" THEN GO EXIT;
        IF Q = "1=0000" THEN
            BEGIN SWITCHIT(PRINTDOLLARBIT); GO AGAIN END;
        IF Q = "1,0000" THEN GO SKANAGAIN;

```

```

02360000 T 0001
02361000 T 0001
02362000 T 0001
02363000 T 0001
02364000 T 0001
02365000 T 0003
02365100 T 0009

```

```

02365200 T 0009
02366000 T 0009
02366100 T 0010
02366200 T 0012
02367000 T 0014
02368000 T 0014
02369000 T 0016
02370000 T 0017
02371000 T 0018
02372000 T 0021
02373000 T 0022
02374000 T 0026
02375000 T 0027
02376000 T 0028
02377000 T 0029
02378000 T 0033

```

```

GO WHATISIT;
LENGTH2: % NO OPTIONS OF THIS LENGTH ARE CURRENTLY IMPLEMENTED.
LENGTH3:
  IF Q = "3SET00" THEN
    BEGIN XMODE:=3; GO SKANAGAIN END;
  IF Q = "3POPO0" THEN
    BEGIN XMODE:=4; GO SKANAGAIN END;
  IF Q = "3NEW00" THEN
    BEGIN
      SWITCHIT(NEWBIT);
      IF Q = "4TAPE0" THEN GO SKANAGAIN;
      GO AGAIN;
    END;
  IF Q = "3SEQ00" THEN
    BEGIN SWITCHIT(SEQBIT); GO AGAIN END;
  IF Q = "3PRT00" THEN
    BEGIN SWITCHIT(PRTBIT); GO AGAIN END;
  IF Q = "3MCP00" THEN
    BEGIN SWITCHIT(MCPBIT); GO AGAIN END;
GO WHATISIT;
LENGTH4:
  IF Q = "4LIST0" THEN
    BEGIN
      SWITCHIT(LISTBIT);
      GO AGAIN;
    END;
  IF Q = "4VOID0" THEN
    BEGIN
      IF XMODE=0 THEN
        BEGIN
          GETVOID(VOIDRANGE, NCR, LCR, INFO[ LASTSEQROW, LASTSEQUENCE]);
          IF VOIDCR=0 THEN VOIDCR:=MKABS(VOIDPLACE) ELSE
            IF COMPARE(MKABS(VOIDRANGE), VOIDCR) # 1 THEN GO TO EXIT;
          MOVE(1, VOIDRANGE, VOIDPLACE);
          GO EXIT;
        END;
      SWITCHIT(VOIDBIT);
      GO AGAIN;
    END;
  IF Q = "4OMIT0" THEN
    BEGIN IF NOT DOLLAR2TOG THEN BEGIN PRINTCARD; FLAG(605); END;
      SWITCHIT(OMITBIT); GO AGAIN END;
  IF Q = "4CARD0" THEN
    BEGIN
      Q1="4TAPE0"; % FAKE OUT SWITCHIT.
      SWITCHIT(MERGE0IT);
      IF XMODE#2 THEN MERGETOG:=NOT MERGETOG;
      OPTIONS[2*MERGE0IT-1] := % CARD IS
        REAL(SETTING & (MERGETOG)[47:1]); % INVERSE OF MERGE.
      IF MERGETOG THEN GO MERGEOPTION;
CARDOPTION:
      LASTUSED:=1;
      GO AGAIN;
    END;
  IF Q = "4TAPE0" THEN
    BEGIN
      SWITCHIT(MERGE0IT);

```

```

02379000 T 0034
02380000 T 0036
02381000 T 0036
02382000 T 0036
02383000 T 0036
02384000 T 0040
02385000 T 0040
02386000 T 0044
02387000 T 0044
02388000 T 0045
02389000 T 0046
02390000 T 0047
02391000 T 0050
02392000 T 0050
02393000 T 0050
02394000 T 0054
02395000 T 0054
02396000 T 0058
02397000 T 0058
02398000 T 0062
02399000 T 0062
02400000 T 0063
02401000 T 0063
02402000 T 0064
02404000 T 0065
02405000 T 0067
02406000 T 0067
02407000 T 0067
02408000 T 0068
02409000 T 0069
02410000 T 0069
02410500 T 0072
02411000 T 0075
02412000 T 0080
02413000 T 0081
02414000 T 0082
02415000 T 0082
02418000 T 0083
02419000 T 0083
02420000 T 0083
02421000 T 0084
02421100 T 0103
02422000 T 0104
02423000 T 0105
02424000 T 0105
02425000 T 0106
02425500 T 0107
02426000 T 0110
02427000 T 0112
02428000 T 0114
02429000 T 0116
02430000 T 0116
02431000 T 0116
02432000 T 0120
02433000 T 0120
02434000 T 0120
02435000 T 0121

```

```

        IF NOT MERGETOG THEN GO CARDOPTION)
MERGEOPTION:
        LASTUSED:=2; % NEXT CARD IS READ FROM READER.
        IF MAXTLCR=0 THEN
            BEGIN
                INTEGER STREAM PROCEDURE FEJ(F,T); VALUE T;

```

PRT(412) = *SEGMENT DESCRIPTOR*

PRT(413) = FEJ

```

        BEGIN
        SI:=F; DI:=LOC T; DS:=WDS;
        SI:=T; SI:=SI+16; DI:=LOC FEJ; DS:=WDS;
        END FEJ;

```

PRT(414) = FIX

STREAM PROCEDURE FIX(F,T); VALUE T;

```

        BEGIN
        SI:=F; SI:=SI+24; DI:=LOC T; DS:=WDS;
        DI:=T; DI:=DI+47; SKIP 4 DB; DS:=2 RESET;
        2(DI:=DI+48); DS:=8 LIT"00#01+0#";
        END FIX;

```

```

        IF GT1:=FEJ(TAPE,0)=10 THEN
            BEGIN
                REWIND(TAPE); FIX(TAPE,0);
            END;
        MAXTLCR:=GT1+TLCR:=9+MKABS(TBUFF[0]);
        READ(TAPE,10,TBUFF[*]); % INITIALIZE TAPE INPUT.
        LASTUSED:=2;
        END;

```

PRT(415) = *SEGMENT DESCRIPTOR*

```

        GO AGAIN;
        END;
        IF Q = "4PAGE0" THEN
            BEGIN
                IF LISTER THEN WRITE(LINE[PAGE]);
                GO SKANAGAIN;
            END;
        IF Q = "4INFO0" THEN
            BEGIN DUMPINFO; GO SKANAGAIN END;
        IF Q = "4SEGS0" THEN
            BEGIN SWITCHIT(SEGSBIT); GO AGAIN END;
        IF Q = "4NEST0" THEN
            BEGIN SWITCHIT(NESTBIT); GO AGAIN END;
        IF Q = "4DECK0" THEN
            BEGIN SWITCHIT(DECKBIT); GO AGAIN END;
        GO WHATISIT;
        LENGTH5;
        IF Q = "5RESET" THEN
            BEGIN XMODE:=2; GO SKANAGAIN END;

```

```

02436000 T 0122
02437000 T 0123
02438000 T 0124
02439000 T 0124
02440000 T 0125
02441000 T 0126

```

START OF SEGMENT ***** 18

```

02442000 T 0000
02443000 T 0000
02444000 T 0000
02445000 T 0001

```

02446000 T 0002

```

02447000 T 0002
02448000 T 0003
02449000 T 0004
02450000 T 0005
02451000 T 0007

```

```

02452000 T 0007
02453000 T 0010
02454000 T 0011
02455000 T 0014
02456000 T 0014
02457000 T 0017
02458000 T 0022
02459000 T 0022

```

18 IS 24 LONG, NEXT SEG 16

```

02460000 T 0128
02461000 T 0128
02462000 T 0128
02463000 T 0129
02464000 T 0129
02465000 T 0134
02466000 T 0136
02467000 T 0136
02468000 T 0136
02469000 T 0140
02470000 T 0140
02471000 T 0144
02472000 T 0144
02473000 T 0148
02474000 T 0148
02475000 T 0152
02476000 T 0152
02477000 T 0153
02478000 T 0153

```



```

IF Q = "5LISTP" THEN
  BEGIN SWITCHIT(LISTPBIT); GO AGAIN; END;
IF Q = "5VOIDT" THEN
  BEGIN
    IF XMODE=0 THEN
      BEGIN
        GETVOID(VOIDRANGE,NCR,LCR,INFO[ LASTSEQROW, LASTSEQUENCE]);
        IF VOIDTCR=0 THEN VOIDTCR:=MKABS(VOIDTPLACE) ELSE
          IF COMPARE(MKABS(VOIDRANGE),VOIDTCR)≠1 THEN GO TO EXIT;
        MOVE(1,VOIDRANGE,VOIDTPLACE);
        GO EXIT;
      END;
    SWITCHIT(VOIDTBIT);
    GO AGAIN;
  END;
IF Q = "5CHECK" THEN
  BEGIN SWITCHIT(CHECKBIT); GO AGAIN; END;
IF Q = "5LIMIT" THEN
  BEGIN
    SKAN;
    IF RESULT≠3 THEN * SHOULD BE NUMBER.
      BEGIN FLAG(600); GO AGAIN; END;
    ERRMAX:=CONV(ACCUM[1],0,ACCUM[1].[12:6]);
    GO SKANAGAIN;
  END;
IF Q = "5PUNCH" THEN
  BEGIN SWITCHIT(PUNCHBIT); GO AGAIN; END;
IF Q = "5PURGE" THEN
  BEGIN SWITCHIT(PURGEBIT); GO AGAIN; END;
IF Q = "5LISTA" THEN
  BEGIN
    SWITCHIT(LISTABIT);
    GO AGAIN;
  END;
IF Q = "5STUFF" THEN
  BEGIN SWITCHIT(STUFFBIT); GO AGAIN; END;
GO WHATISIT;
LENGTH6;
IF Q = "6SEQR" THEN
  BEGIN SWITCHIT(SEQERRBIT); GO AGAIN; END;
IF Q = "6SINGL" THEN
  BEGIN SWITCHIT(SINGLBIT); GO AGAIN; END;
IF Q = "6SEQXE" THEN
  BEGIN
    SEQXEQTOG:=XMODE≠2 AND XMODE≠4 OR SEQXEQTOG;*NEVER RESET.
    IF BUILDLINE.[45:1] THEN BUILDLINE:=TRUE;
    GO SKANAGAIN;
  END;
IF Q = "6DEBUG" THEN
  BEGIN
    SWITCHIT(DEBUGBIT);
    IF DEBUGTOG THEN
      IF WOP[0]≠0 THEN
        BEGIN
          FILL WOP[*] WITH
            "LITC"," ",

```

```

02479000 T 0157
02480000 T 0157
02481000 T 0161
02482000 T 0161
02483000 T 0162
02484000 T 0163
02485000 T 0163
02485500 T 0166
02486000 T 0169
02487000 T 0174
02488000 T 0175
02489000 T 0176
02490000 T 0176
02493000 T 0177
02494000 T 0177
02495000 T 0177
02496000 T 0178
02497000 T 0181
02498000 T 0181
02499000 T 0182
02500000 T 0186
02501000 T 0186
02502000 T 0190
02503000 T 0193
02504000 T 0193
02505000 T 0193
02506000 T 0194
02507000 T 0198
02508000 T 0198
02509000 T 0202
02510000 T 0202
02511000 T 0203
02513000 T 0204
02514000 T 0206
02515000 T 0206
02516000 T 0206
02517000 T 0210
02518000 T 0210
02519000 T 0211
02520000 T 0211
02521000 T 0215
02522000 T 0215
02523000 T 0219
02524000 T 0219
02525000 T 0220
02526000 T 0223
02527000 T 0225
02528000 T 0227
02529000 T 0227
02530000 T 0227
02531000 T 0228
02533000 T 0229
02534000 T 0229
02535000 T 0231
02536000 T 0231
02537000 T 0232

```

```

      "OPDC", "DESC",
11, "NOP ", 12, "PRT ", 13, "DEL ", 16, "ADD ", 18, "PRL ", 19, "LNG ",
21, "GEQ ", 22, "BBC ", 24, "INX ", 35, "LOR ", 37, "GTR ", 38, "BFC ",
39, "RTN ", 40, "CDC ", 48, "SUB ", 64, "MUL ", 67, "LND ", 68, "STD ",
69, "NEQ ", 71, "XIT ", 72, "MKS ", 128, "DIV ", 130, "COM ", 131, "LQV ",
132, "SND ", 133, "XCH ", 134, "CHS ", 167, "RTS ", 168, "CDC ", 260, "LOD ",
261, "DUP ", 278, "LBC ", 294, "LFC ", 322, "ZP1 ", 384, "IDV ", 532, "ISD ",
533, "LEQ ", 534, "BBW ", 548, "ISN ", 549, "LSS ", 550, "BFW ", 581, "EQL ",
582, "SSP ", 790, "LBU ", 806, "LFU ", 896, "RDV ",
1003, "SCI ", 1004, "SAN ", 1019, "SCS ",
1023, 1023, 1023, 1023, 1023, 1023, 1023, 1023, 1023, 1023, 1023, 1023, 1023,
1023, 1023, 1023, 1023, 1023, 1023, 1023, 1023, 1023, 1023, 1023, 1023,

```

```

0, " ", 0, 0,
      FILL COP[*] WITH % CHARACTER MODE MNEMONICS

```

```

0, "EXC ", 2, "BSD ", 3, "BSS ", 4, "RDA ", 5, "TRW ", 6, "SED ", 7, "TDA ",
12, "SDA ", 13, "SSA ", 14, "SFD ", 15, "SRD ", 18, "SES ", 20, "TEQ ", 21, "TNE ",
22, "TEG ", 23, "TGR ", 24, "SRS ", 25, "SFS ", 28, "TEL ", 29, "TLS ", 30, "TAN ",
31, "BIT ", 32, "INC ", 33, "STC ", 34, "SEC ", 35, "CRF ", 36, "JNC ", 37, "JFC ",
38, "JNS ", 39, "JFW ", 40, "RCA ", 41, "ENS ", 42, "BNS ", 43, "RSA ", 44, "SCA ",
45, "JRC ", 46, "TSA ", 47, "JRV ", 48, "CEQ ", 49, "CNE ", 50, "CEG ", 51, "CGR ",
52, "BIT ", 53, "BIR ", 54, "OCV ", 55, "ICV ", 56, "CEL ", 57, "CLS ", 58, "FSU ",
59, "FAD ", 60, "TRP ", 61, "TRN ", 62, "TRZ ", 63, "TRS ", 64, 0, 64, 0, 64, 0, 64, 0,
64, 0, 64, 0, 64, 0, 64, 0;

```

```

      END;
      GO AGAIN;
      END;
      IF Q = "6FORMA" THEN
        BEGIN SWITCHIT(FORMATBIT); GO AGAIN; END;
      GO WHATISIT;
LENGTH7:
% IF Q = "7INCLU" THEN
% BEGIN DOLLARCARD:=STARTINCLUDING; GO EXIT; END;
% IF Q = "7INCLN" THEN
% BEGIN SWITCHIT(NEWINCLBIT); GO AGAIN; END;
LENGTH8: % NO OPTIONS OF THIS LENGTH ARE CURRENTLY IMPLEMENTED.
LENGTH9:
      IF Q = "9INTRI" THEN
        BEGIN
          INTOG:=XMODE*2 AND XMODE*4 OR INTOG; % NEVER RESET.
          GO SKANAGAIN;
        END;
      WHATISIT:
      IF RESULT=3 THEN
        BEGIN
          BASENUM:=CONV(ACCUM[1],0,ACCUM[1].[12:6]);
          TOTALNO:=10;
          NEWBASE:=TRUE;
          GO SKANAGAIN;
        END;
      IF RESULT=2 THEN
        BEGIN
          IF Q = "1+0000" THEN
            BEGIN
              SKAN;
            END;
          END;
        END;
      END;

```

```

02538000 T 0233
02539000 T 0233
02540000 T 0233
02541000 T 0233
02542000 T 0233
02543000 T 0233
02544000 T 0233
02545000 T 0233
02546000 T 0233
02547000 T 0233
02548000 T 0233
02549000 T 0233
19 IS 128 LONG, NEXT SEG 16
02550000 T 0233
02551000 T 0234
START OF SEGMENT ***** 20
02552000 T 0235
02553000 T 0235
02554000 T 0235
02555000 T 0235
02556000 T 0235
02557000 T 0235
02558000 T 0235
02559000 T 0235
02560000 T 0235
20 IS 128 LONG, NEXT SEG 16
02563000 T 0235
02564000 T 0235
02565000 T 0237
02566000 T 0237
02567000 T 0237
02568000 T 0241
02569000 T 0241
02570000 T 0242
02571000 T 0242
02572000 T 0242
02573000 T 0242
02574000 T 0242
02575000 T 0242
02576000 T 0242
02576500 T 0242
02577000 T 0243
02577250 T 0247
02577500 T 0249
02578000 T 0249
02579000 T 0249
02580000 T 0249
02581000 T 0250
02582000 T 0253
02583000 T 0254
02584000 T 0255
02585000 T 0255
02586000 T 0255
02587000 T 0256
02588000 T 0257
02589000 T 0257
02590000 T 0258

```

```

        IF RESULT=3 THEN
            ADDVALUE:=CONV(ACCUM[1],0,ACCUM[1],[12:6])
        ELSE FLAG(600); % NUMBER EXPECTED.
        END;
        GO SKANAGAIN;
    END;
COMMENT DID NOT RECOGNIZE OPTION;
    IF RESULT#1 THEN % NOT AN IDENTIFIER.
        BEGIN FLAG(601); GO SKANAGAIN END;
    SWITCHIT(USEROPINX); % USEROPINX MEANS A USER-DEFINED OPTION.
    GO AGAIN;
EXIT;
    LISTER:=DEBUGTOG OR LISTOG OR LISTATOG;
    MOVE(10,DEFINEARRAY[0],ACCUM[0]); % RESTORE INFORMATION FOR
    COUNT:=SCOUNT; RESULT:=SRESULT; % "TABLE" TO RESUME SCAN.
    RESTORESEQNUM(LCR,INFO[LASTSEQROW, LASTSEQUENCE]); % FOR VOID TESTS
    DOLLARTOG:=FALSE;
    END DOLLARCARD;

```

```

02591000 T 0262
02592000 T 0262
02593000 T 0265
02594000 T 0268
02595000 T 0268
02596000 T 0269
02597000 T 0269
02598000 T 0269
02599000 T 0270
02600000 T 0271
02601000 T 0272
02602000 T 0273
02602500 T 0273
02602600 T 0276
02602700 T 0278
02602800 T 0279
02603000 T 0281
02604000 T 0282
16 IS 285 LONG, NEXT SEG 3

```

```

COMMENT TABLE IS THE ROUTINE THAT MOST CODE IN THE COMPILER
USES WHEN IT IS DESIRED TO SCAN ANOTHER LOGICAL QUANTITY.
THE RESULT RETURNED IS THE CLASS OF THE ITEM DESIRED.
TABLE MAINTAINS THE VARIABLES I AND NXTELBT AND THE ARRAY
ELBAT. ELBAT AND I ARE PRINCIPAL VARIABLES USED FOR
COMMUNICATION BETWEEN TABLE AND THE OUTSIDE WORLD. NXTELBT
IS ALMOST EXCLUSIVELY USED BY TABLE, ALTHOUGH AN OCCASION-
AL OTHER USE IS MADE IN ORDER TO FORGET THAT SOMETHING WAS
SCANNED. (SEE, FOR EXAMPLE, COMPOUNDTAIL). FOR FURTHER
GENERAL DISCUSSION SEE THE DECLARATION OF THESE VARIABLES.
THE PARAMETER P IS THE ACTUAL INDEX OF THE QUANTITY
DESIRED (USUALLY I-1, I, OR I+1).
THE GENERAL PLAN OF TABLE IS THIS:
    I) IF P < NXTELBT GO ON TO III).
    II) PROCESS ONE QUANTITY.
        A) SCAN.
        B) TEST FOR IDENTIFIER, NUMBER, OR SPECIAL CHARACTER.
            1) IDENTIFIER = LOOKUP IN DIRECTORY AND PROCESS
              IN SPECIAL MANNER IF COMMENT OR DEFINED ID.
            2) NUMBER = PROCESS INTEGER PART, FRACTIONAL PART,
              AND EXPONENT PART.
            3) TEST IF SPECIAL CHARACTER REQUIRES SPECIAL
              PROCESSING = OTHERWISE GET ELBAT WORD FROM
              SPECIAL.
        C) LOAD ELBAT AND INCREMENT NXTELBT.
        D) IF ELBAT IS FULL ADJUST ELBAT, NXTELBT, I, AND P.
        E) GO BACK TO I).
    III) RETURN WITH CLASS OF ELBAT[P].
    FURTHER DETAILS ARE GIVEN IN BODY OF TABLE.

```

```

02605000 T 0205
02606000 T 0205
02607000 T 0205
02608000 T 0205
02609000 T 0205
02610000 T 0205
02611000 T 0205
02612000 T 0205
02613000 T 0205
02614000 T 0205
02615000 T 0205
02616000 T 0205
02617000 T 0205
02618000 T 0205
02619000 T 0205
02620000 T 0205
02621000 T 0205
02622000 T 0205
02623000 T 0205
02624000 T 0205
02625000 T 0205
02626000 T 0205
02627000 T 0205
02628000 T 0205
02629000 T 0205
02630000 T 0205
02631000 T 0205
02632000 T 0205
02633000 T 0205
02634000 T 0205
02635000 T 0205

```

```

;
INTEGER PROCEDURE TABLE(P); VALUE P; INTEGER P;
PRT(416) = TABLE

```

```

BEGIN
    LABEL PERCENT, SPECIALCHAR, COMPLETE, COLON, DOT, ATSIGN, QUOTE,

```

```

02636000 T 0205
02637000 T 0205

```

	START OF SEGMENT *****	21
STRNGXT,MOVEIT,ARGH,FINISHNUMBER,	02638000 T	0000
SCANAGAIN,FPART,EPART,IPART,IDENT,ROSE,COMPOST,DOLLAR,RTPAREN,	02639000 T	0000
CROSSHATCH,NUMBEREND;	02640000 T	0000
SWITCH SPECIALSWITCH:=PERCENT,DOLLAR,DOT,ATSIGN,COLON,QUOTE,	02641000 T	0000
RTPAREN,CROSSHATCH;	02642000 T	0002
SWITCH RESULTSWITCH:=IDENT,SPECIALCHAR,IPART;	02643000 T	0007
WHILE P ≥ NXTELBT	02644000 T	0012
DO BEGIN	02645000 T	0013
SCANAGAIN:	02646000 T	0014
COUNT:=RESULT:=ACCUM[1]:=0; SCANNER;	02647000 T	0015
GO RESULTSWITCH[RESULT];	02648000 T	0017
ARGH:	02649000 T	0019
Q:=ACCUM[1]; FLAG(141); GO SCANAGAIN;	02650000 T	0020
SPECIALCHAR:	02651000 T	0022
GT1:=ACCUM[1],[18:6] = 2;	02652000 T	0023
ENDTOG:=GT1 = 57 AND ENDTOG;	02653000 T	0025
COMMENT OBTAIN ACTUAL CHARACTER FROM ACCUM;	02654000 T	0026
T:=SPECIAL[GT1>1[42:41:3]];	02655000 T	0026
COMMENT NOTICE COMPRESSION TECHNIQUE USED TO SHORTEN TABLE OF	02656000 T	0028
ELBAT WORDS FOR SPECIAL CHARACTERS;	02657000 T	0028
IF GT1:=T.INCR = 0 THEN GO COMPLETE;	02658000 T	0028
GO SPECIALSWITCH[GT1];	02659000 T	0031
COMMENT INCR FIELD OF SPECIAL CHARACTER IS NON-ZERO FOR SPECIAL	02660000 T	0033
CHARACTERS REQUIRING SPECIAL HANDLING. INCR IS SWITCHED	02661000 T	0033
ON TO OBTAIN DISCRIMINATION;	02662000 T	0033
COLON: RESULT:=7; SCANNER; COMMENT ELIMINATE BLANKS - CHECKING	02663000 T	0033
FOR := IN PLACE OF + ;	02664000 T	0034
IF EXAMIN (NCR) = "=" THEN	02665000 T	0034
BEGIN RESULT:=0; SCANNER; T:=SPECIAL[13] END;	02666000 T	0036
RESULT:=2; GO COMPLETE;	02667000 T	0038
DOT:	02668000 T	0040
IF EXAMIN(NCR)>9 OR ENDTOG THEN GO COMPLETE;	02680000 T	0040
NH1:=NL0:=0;	02681000 T	0043
C:=0; GO FPART;	02682000 T	0044
ATSIGN:	02683000 T	0045
RESULT:=0; SCANNER; % SCAN PAST "@",	02684000 T	0046
IF COUNT>17 THEN GO ARGH; % 16 CHARS. + "@",	02685000 T	0047
IF OCTIZE(ACCUM[1],C,17-COUNT,COUNT=1) THEN	02686000 T	0048
BEGIN Q:=ACCUM[1]; FLAG(521); GO SCANAGAIN END;	02686500 T	0051
GO NUMBEREND;	02687000 T	0054
COMMENT DOT AND ATSIGN ENTER NUMBER CONVERSION AT CORRECT SPOT;	02689000 T	0055
QUOTE:	02690000 T	0055
COUNT:=0;	02691000 T	0055
T:=IF STREAMTOG THEN 63	02692000 T	0055
ELSE IF REAL(STREAMTOG)>1 THEN 8 ELSE 7;	02692500 T	0056
DO BEGIN	02693000 T	0060
RESULT:=5; SCANNER;	02694000 T	0060
IF COUNT>T THEN	02695000 T	0061
BEGIN Q:=ACCUM[1]; FLAG(520); GO SCANAGAIN END;	02696000 T	0062
END UNTIL EXAMIN(NCR) = "=";	02697000 T	0064
Q:=ACCUM[1]; RESULT:=5; SCANNER; COUNT:=COUNT-1;	02698000 T	0067
IF COUNT<0 THEN COUNT:=COUNT+64;	02699000 T	0070
ACCUM[1]:=Q; RESULT:=4;	02700000 T	0073
STRNGXT: T:=C:=0;	02701000 T	0075
IF COUNT < 8 THEN	02703000 T	0076
MOVEIT:	02704000 T	0077

```

MOVECHARACTERS(COUNT,ACCUM[1],3,C,8-COUNT);
T,CLASS:=STRNGCON;
GO COMPLETE;
COMMENT CROSSHATCH HANDLES TWO SITUATIONS:
THE CROSSHATCH AT END OF DEFINE DECLARATIONS AND
THE CROSSHATCH AT END OF ALPHA REPRESENTING DEFINED IDS.
THE TWO CASES ARE PROCESSED DIFFERENTLY. THE FIRST CASE
MERELY PLACES THE CROSSHATCH IN ELBAT. THE SECOND CASE
CAUSES AN EXIT FROM SCANNING THE ALPHA FOR THE DEFINED ID.
FOR A FULL DISCUSSION SEE DEFINEGEN;
CROSSHATCH:
IF DEFINECTR#0 THEN GO COMPLETE;
PUTSEQNO(GT1,LCR);
TURNONSTOPLIGHT(0,LCR);
IF DEFINEINDEX = 0 THEN GO ARGH;
LCR:=(GT1:=DEFINEARRAY[DEFINEINDEX=1]) DIV 262144;
NCR:=GT1 MOD 262144;
GT2:=0&(T:=DEFINEARRAY[DEFINEINDEX:=DEFINEINDEX=3])[33:18:15];
LASTUSED:=T,[33:15];
FOR GT1:=1 STEP 1 UNTIL GT2 DO
    BEGIN
        STACKHEAD[(T:=TAKE(LASTINFO+1)),[12:36] MOD 125]:=
            TAKE(LASTINFO).LINK;
        LASTINFO:=(NEXTINFO:=LASTINFO)=T.PURPT;
    END;
GO SCANAGAIN;
DOLLAR: COMMENT THIS CODE HANDLES CONTROL CARDS;
DOLLARCARD;
PERCENT: IF NCR # FCR THEN READACARD;
GO SCANAGAIN;
COMMENT MOST PERCENT SIGNS ACTING AS END OF CARD SENTINELS GET TO
PERCENT. PERCENT READS THE NEXT CARD AND STARTS OVER. A
SIDE EFFECT IS THAT ALL CHARACTERS ON A CARD ARE IGNORED
AFTER A FREE PERCENT SIGN (ONE NOT IMBEDDED IN A STRING OR
COMMENT);
COMMENT MIGHT BE FUNNY COMMA = HANDLE HERE;
RTPAREN: RESULT:=7; SCANNER;
IF EXAMIN(NCR) = "" THEN
    BEGIN
        RESULT:=0; SCANNER;
        DO BEGIN
            RESULT:=5; SCANNER
            END UNTIL EXAMIN(NCR) = "";
        RESULT:=0; SCANNER;
        RESULT:=7; SCANNER;
        IF EXAMIN(NCR) # "(" THEN GO ARGH;
        RESULT:=0; SCANNER; Q:=ACCUM[1];
        T:=SPECIAL[24]
    END;
RESULT:=2; GO COMPLETE;
IPART: TCOUNT:=0; C:=CONVERT;
% RESULT:=7; SCANNER; % DEBLANK,
% IF DEFINECTR#0 THEN
% IF (C#3 OR C#4) AND EXAMIN(NCR)="" THEN %OCTAL OR HEX STRING.
% BEGIN INTEGER SIZ;
% RESULT:=5; SCANNER; % SKIP QUOTE,
% COUNT:=0;

```

```

02705000 T 0078
02705100 T 0080
02705200 T 0082
02707000 T 0082
02708000 T 0082
02709000 T 0082
02710000 T 0082
02711000 T 0082
02712000 T 0082
02713000 T 0082
02714000 T 0082
02715000 T 0083
02716000 T 0084
02717000 T 0085
02718000 T 0086
02719000 T 0087
02720000 T 0090
02721000 T 0091
02722000 T 0094
02723000 T 0095
02723500 T 0098
02724000 T 0098
02725000 T 0101
02726000 T 0102
02727000 T 0105
02728000 T 0107
02729000 T 0107
02730000 T 0108
02731000 T 0108
02737000 T 0110
02738000 T 0111
02739000 T 0111
02740000 T 0111
02741000 T 0111
02742000 T 0111
02743000 T 0111
02744000 T 0111
02745000 T 0113
02746000 T 0115
02747000 T 0115
02748000 T 0116
02749000 T 0117
02750000 T 0117
02751000 T 0120
02752000 T 0121
02753000 T 0123
02754000 T 0125
02755000 T 0127
02756000 T 0127
02757000 T 0128
02758000 T 0129
02759000 T 0131
02760000 T 0131
02761000 T 0131
02762000 T 0131
02763000 T 0131
02764000 T 0131

```

```

% DO BEGIN
% RESULT:=5; SCANNER;
% IF COUNT > SIZE:=48 DIV C THEN % > 1 WORD LONG.
% BEGIN ERR(520); GO SCANAGAIN END;
% END UNTIL EXAMIN(NCR)="" ;
% Q:=ACCUM[1]; RESULT:=5; SCANNER; COUNT:=COUNT+1;
% IF C=3 THEN % OCTAL STRING.
% IF OCTIZE(ACCUM[1],ACCUM[4],16-COUNT,COUNT) THEN
% FLAG(521) % NON OCTAL CHARACTER IN STRING.
% ELSE ELSE IF HEXIZE(ACCUM[1],ACCUM[4],12-COUNT,COUNT) THEN
% FLAG(521); % NON CHARACTER IN HEX STRING.
% IF COUNT < SIZE THEN
% BEGIN
% C:=ACCUM[4]; GO FINISHNUMBER;
% END;
% T.INCR:=COUNT:=8; T.CLASS:=STRING;
% MOVECHARACTERS(8,ACCUM[4],0,ACCUM[1],3);
% GO COMPLETE;
% END OCTAL OR HEX STRING;
% IF DPTOG THEN
% BEGIN NHI:=THI; NLO:=TLO; END;
% IF EXAMIN(NCR)=",." THEN
% BEGIN
% RESULT:=0; SCANNER;
% C:=1.0x C;
FPART: TCOUNT:=COUNT;
% IF EXAMIN(NCR)<9 THEN
% BEGIN
% RESULT:=0; SCANNER;
% IF DPTOG THEN
% BEGIN
% DOUBLE(CONVERT,TLO,TEN[(COUNT-TCOUNT)MOD 12],
% 0,/,:=,THI,TLO);
% FOR T:=12 STEP 12 UNTIL COUNT = TCOUNT DO
% DOUBLE(THI,TLO,TEN[12],0,/,:=,THI,TLO);
% DOUBLE(THI,TLO,NHI,NLO,+,:=,NHI,NLO);
% C:=NHI
% END
% ELSE C:=TEN[TCOUNT-COUNT]*CONVERT+C;
% END
% END;
% RESULT:=7; SCANNER;
% IF EXAMIN(NCR)="#" THEN
% BEGIN
% RESULT:=0; SCANNER;
EPART: TCOUNT:=COUNT;
% C:=C*1.0;
% RESULT:=7; SCANNER;
% IF T:=EXAMIN(NCR)>9 THEN
% BEGIN
% RESULT:=0; SCANNER;
% TCOUNT:=COUNT;
% END;
% RESULT:=0; SCANNER;
% Q:=ACCUM[1];
% IF GT1:=T:=(IF T="#" THEN =CONVERT ELSE CONVERT)<=46 OR
% T>69 THEN FLAG(269)

```

```

02765000 T 0131
02766000 T 0131
02767000 T 0131
02768000 T 0131
02769000 T 0131
02770000 T 0131
02771000 T 0131
02772000 T 0131
02773000 T 0131
02774000 T 0131
02775000 T 0131
02776000 T 0131
02777000 T 0131
02778000 T 0131
02779000 T 0131
02780000 T 0131
02781000 T 0131
02782000 T 0131
02783000 T 0131
02784000 T 0131
02785000 T 0132
02786000 T 0134
02787000 T 0135
02788000 T 0136
02789000 T 0137
02790000 T 0138
02791000 T 0139
02792000 T 0141
02793000 T 0142
02794000 T 0143
02795000 T 0143
02796000 T 0144
02797000 T 0147
02798000 T 0148
02799000 T 0153
02800000 T 0158
02801000 T 0160
02802000 T 0160
02803000 T 0161
02804000 T 0164
02805000 T 0164
02806000 T 0164
02807000 T 0166
02808000 T 0167
02809000 T 0168
02810000 T 0169
02811000 T 0170
02812000 T 0172
02813000 T 0173
02815000 T 0175
02816000 T 0176
02817000 T 0177
02818000 T 0178
02820000 T 0178
02822000 T 0179
02823000 T 0180
02824000 T 0185

```

```

ELSE BEGIN
    T:=TEN[T];
    IF ABS(0&C[42:3:6]&C[1:2:1]+0&T[42:3:6]&T[1:2:1]
        + 12) > 63 THEN FLAG(269)
ELSE IF DPTOG THEN
    IF GT1<0 THEN
        BEGIN
            GT1:=GT1;
            DOUBLE(NHI,NLO,TEN[GT1 MOD 12],0,/,/,NHI,NLO);
            FOR GT2:=12 STEP 12 UNTIL GT1 DO
                DOUBLE(NHI,NLO,TEN[12],0,/,/,NHI,NLO);
            END
        ELSE BEGIN
            DOUBLE(NHI,NLO,TEN[GT1 MOD 12],0,x,/,NHI,NLO);
            FOR GT2:=12 STEP 12 UNTIL GT1 DO
                DOUBLE(NHI,NLO,TEN[12],0,x,/,NHI,NLO);
            END
        ELSE C:=C×T;
        END;
    END;
NUMBEREND:
    Q:=ACCUM[1]; RESULT:=3;
FINISHNUMBER:
    T:=0;
    IF C.[1:37]=0 THEN
        BEGIN T.CLASS:=LITNO ; T.ADDRESS:=C END
    ELSE T.CLASS:=NONLITNO ;
    GO COMPLETE;
COMMENT THE CODE BETWEEN IDENT AND COMPOST DOES A LOOKUP IN INFO,
        IF QUANTITY IS NOT FOUND THE ELBAT WORD EXPECTS TO BE
        ZERO, THE SCRAMBLE FOR APPROPRIATE STACK IS FIRST THING
        TO BE DONE, THEN THE LOOP BETWEEN COMPOST AND
        ROSE IS ENTERED, THE LAST THING DONE FOR ANY
        IDENTIFIER WHICH IS FOUND IS TO STUFF THE LOCATION
        OF THE ELBATWORD IN INFO INTO THE LINK FIELD, THIS
        ALLOWS REFERENCE BACK TO INFO FOR ADDITIONAL DATA,
        SHOULD THIS BE REQUIRED.
IDENT: T:=STACKHEAD[SCRAM:=(Q:=ACCUM[1])MOD 125];
ROSE: GT1:=T.LINKR;
        IF(GT2:=T.LINKC)+GT1= 0 THEN
            BEGIN T:=0; GO COMPLETE END;
        IF T = INFO[GT1, GT2] THEN BEGIN
            T:= 0; GO TO COMPLETE END;
            T:=INFO[GT1,GT2];
            IF INFO[GT1,GT2+1]&0[1:1:11] ≠ 0 THEN GO ROSE;
            IF COUNT ≤ 5 THEN GO COMPOST ;
            IF NOT EQUAL(COUNT=5,ACCUM[2],INFO[GT1,GT2+2])THEN GO ROSE;
COMPOST: T:=T&GT1[35:43:5]&GT2[40:40:8];
COMMENT CHECK HERE FOR COMMENTS AND DEFINED IDS;
        IF NOT ENDTOG THEN
            BEGIN
                IF GT1:=T.CLASS = COMMENTV THEN
                    BEGIN
                        WHILE EXAMIN(NCR) ≠ "J" DO
                            BEGIN RESULT:=6; COUNT:=0; SCANNER END;
                        RESULT:=0;SCANNER;GO SCANAGAIN
                    END

```

02825000	T	0186
02826000	T	0189
02827000	T	0190
02828000	T	0195
02829000	T	0197
02830000	T	0198
02831000	T	0199
02832000	T	0200
02833000	T	0201
02834000	T	0205
02835000	T	0206
02836000	T	0211
02837000	T	0211
02838000	T	0212
02839000	T	0215
02840000	T	0217
02841000	T	0222
02842000	T	0222
02843000	T	0224
02844000	T	0224
02845000	T	0224
02846000	T	0225
02847000	T	0226
02848000	T	0227
02849000	T	0227
02850000	T	0229
02851000	T	0233
02852000	T	0235
02853000	T	0236
02854000	T	0236
02855000	T	0236
02859000	T	0236
02860000	T	0236
02861000	T	0236
02862000	T	0236
02863000	T	0236
02864000	T	0236
02865000	T	0236
02875000	T	0239
02876000	T	0241
02877000	T	0243
02877010	C	0245
02877020	C	0247
02878000	T	0248
02879000	T	0250
02880000	T	0254
02881000	T	0255
02882000	T	0260
02883000	T	0262
02884000	T	0262
02885000	T	0263
02886000	T	0263
02887000	T	0265
02888000	T	0266
02889000	T	0268
02890000	T	0270
02891000	T	0272

```

END;
IF STOPDEFINE THEN GO COMPLETE;
IF GT1 ≠ DEFINEDID THEN GO COMPLETE;
COMMENT SETUP FOR DEFINED IDS - SEE DEFINEGEN FOR MORE DETAILS;
IF T.ADDRESS≠0 THEN T:=FIXDEFINEINFO(T);
IF DEFINEINDEX = 24 THEN
    BEGIN FLAG(139);GO ARGH END;
DEFINEARRAY[DEFINEINDEX]:=LASTUSED&T.ADDRESS [18:33:15];
LASTUSED:=GIT(T);
DEFINEARRAY[DEFINEINDEX+2]:=262144×LCR+NCR;
LCR:=(NCR:=MKABS(DEFINEARRAY[DEFINEINDEX+1]))+1;
PUTSEQNO(GT4,LCR);
TURNONSTOPLIGHT("X",LCR); DEFINEINDEX:=DEFINEINDEX+3;
GO PERCENT;
COMPLETE:
ELBAT[NXTELBT]:=T;
STOPDEFINE:=FALSE; COMMENT ALLOW DEFINES AGAIN;
IF NXTELBT:=NXTELBT+1 > 74 THEN
    IF NOT MACROID THEN
        BEGIN
COMMENT ELBAT IS FULL; ADJUST IT;
MOVE(10,ELBAT[65],ELBAT);
I:=I-65; P:=P-65; NXTELBT:=10;
END
END; % OF DO WHILE P≥NXTELBT
IF TABLE:=ELBAT[P].CLASS = COMMENTV THEN
    BEGIN
COMMENT SPECIAL HANDLING OF CONSTANTS FOR SAKE OF FOR STATEMENTS;
C:=INFO[0,ELBAT[P].ADDRESS];
ELBAT[P].CLASS:=TABLE:=NONLITNO
END;
STOPDEFINE:=FALSE; COMMENT ALLOW DEFINES;
END TABLE ;

```

← apparently
redundant
code.

?

←

```

02892000 T 0272
02893000 T 0272
02894000 T 0273
02895000 T 0274
02896000 T 0274
02898000 T 0277
02899000 T 0278
02900000 T 0280
02901000 T 0282
02902000 T 0283
02903000 T 0286
02904000 T 0290
02905000 T 0291
02906000 T 0293
02909000 T 0295
02910000 T 0295
02911000 T 0296
02912000 T 0297
02913000 T 0298
02914000 T 0299
02915000 T 0300
02916000 T 0300
02917000 T 0302
02918000 T 0305
02919000 T 0305
02920000 T 0305
02921000 T 0309
02922000 T 0309
02923000 T 0309
02924000 T 0312
02925000 T 0313
02926000 T 0315
02927000 T 0315

```

21 IS 321 LONG, NEXT SEG 3

```

PRT(417) = BOOLPRIM
PRT(420) = BOOLCOMP
PRT(421) = NEXT

BOOLEAN PROCEDURE BOOLPRIM; FORWARD;
PROCEDURE BOOLCOMP(B); BOOLEAN B; FORWARD;
INTEGER PROCEDURE NEXT;
    BEGIN
    LABEL EXIT;
    INTEGER T;
    DEFINE ERROR = BEGIN FLAG(603); GO EXIT END;
    SKAN;
    IF RESULT=3 THEN ERROR; % NUMBERS NOT ALLOWED.
    IF RESULT=2 THEN % SPECIAL CHARACTER.
        BEGIN
            T:=IF Q="1,0000" OR Q="1%0000" THEN 20 % FAKE OUT BOULEXP.
            ELSE ((T:=Q.[18:6]-2) & T[42:41:13]);
            IF T=11 OR T=19 OR T=20 THEN BATMAN:=SPECIALIT % (,),UR ;

```

```

02955000 T 0205
02955500 T 0205
02956000 T 0205
02956500 T 0205
02957000 T 0205
START OF SEGMENT ***** 22
02957500 T 0000
02958000 T 0000
02958500 T 0000
02959000 T 0003
02959500 T 0006
02960000 T 0007
02960500 T 0007
02961000 T 0010
02961500 T 0013

```



```

        ELSE FLAG(603);
        GO EXIT
        END SPECIAL CHARACTERS;
COMMENT LOOK FOR BOOLEAN OPERATORS, THEN OPTIONS;
T:=      IF Q="3NOT00" THEN NOTOP
        ELSE IF Q="3AND00" THEN ANDOP
        ELSE IF Q="2OR000" THEN OROP
        ELSE IF Q="3EQV00" THEN EQVOP
        ELSE 0;
        IF T#0 THEN BATMAN.CLASS:=T
ELSE BATMAN:=1 & BOOID(2:7) & REAL(FINDOPTION(1))[1:1]; % OPTION.
EXIT;
        NEXT:=MYCLASS:=BATMAN.CLASS;
END NEXT;

```

```

02962000 T 0017
02962500 T 0021
02963000 T 0022
02963500 T 0022
02964000 T 0022
02964500 T 0023
02965000 T 0025
02965500 T 0027
02966000 T 0029
02966500 T 0031
02967000 T 0032
02967500 T 0042
02968000 T 0043
02968500 T 0044
22 IS 48 LONG, NEXT SEG 3

```

```

        BOOLEAN PROCEDURE BOOLEXP;
        BEGIN
        BOOLEAN B;

STACK(F+3) = B

        B:=BOOLPRIM;
        WHILE MYCLASS#EQVOP AND MYCLASS#ANDOP DO BOULCOMP(B);
        BOOLEXP:=B
        END BOOLEXP;

```

```

02969000 T 0205
02969500 T 0205
02970000 T 0205
START OF SEGMENT ***** 23
02970500 T 0000
02971000 T 0001
02971500 T 0004
02972000 T 0004
23 IS 8 LONG, NEXT SEG 3

```

```

        BOOLEAN PROCEDURE BOOLPRIM;
        BEGIN
        BOOLEAN B,KNOT;

STACK(F+3) = B
STACK(F+4) = KNOT

        DEFINE SKIPIT = MYCLASS:=NEXT #;
        IF KNOT:=(NEXT=NOTOP) THEN SKIPIT;
        IF MYCLASS=LEFTPAREN THEN
        BEGIN
        B:=BOOLEXP;
        IF MYCLASS#RTPAREN THEN FLAG(604);
        END
ELSE IF MYCLASS#BOOID THEN FLAG(601)
ELSE B:=BATMAN<0;
        IF KNOT THEN B:=NOT B; SKIPIT;
        BOOLPRIM:=B
        END BOOLPRIM;

```

```

02972500 T 0205
02973000 T 0205
02973500 T 0205
START OF SEGMENT ***** 24
02974000 T 0000
02974500 T 0000
02975000 T 0003
02975500 T 0003
02976000 T 0004
02976500 T 0005
02977000 T 0007
02977500 T 0007
02978000 T 0009
02978500 T 0011
02979000 T 0014
02979500 T 0014
24 IS 18 LONG, NEXT SEG 3

```

```

PROCEDURE BOULCOMP(B); BOOLEAN B;

```

```

02980000 T 0205

```

```

                BEGIN
                REAL OPCLASS;

STACK(F+2) = OPCLASS
                BOOLEAN T;

STACK(F+3) = T
                OPCLASS:=MYCLASS;
                T:=BOOLPRIM;
                WHILE OPCLASS<MYCLASS DO BOOLCOMP(T);
                B:=      IF OPCLASS=ANDOP THEN (B AND T)
                        ELSE IF OPCLASS=OROP  THEN (B OR T)
                        ELSE
                                (B EQV T);
                END BOOLCOMP;

```

```

02980500 T 0205
02981000 T 0205
START OF SEGMENT ***** 25
02981500 T 0000
02982000 T 0000
02982500 T 0000
02983000 T 0001
02983500 T 0004
02984000 T 0006
02984500 T 0008
02985000 T 0011
25 IS 14 LONG, NEXT SEG 3

```

```

%
COMMENT#####
                FORWARD DECLARATIONS
#####;
%
PRT(422) = AEXP      PROCEDURE AEXP; FORWARD;
PRT(423) = ARITHSEC  PROCEDURE ARITHSEC; FORWARD;
PRT(424) = SIMPARITH PROCEDURE SIMPARITH; FORWARD;
PRT(425) = ARITHCOMP PROCEDURE ARITHCOMP; FORWARD;
PRT(426) = PRIMARY  PROCEDURE PRIMARY; FORWARD;
                DEFINE BEXP = AEXP#;
                INTEGER PROCEDURE EXPRSS; FORWARD;
PRT(427) = EXPRSS    PROCEDURE POLISHER(EXPECT); VALUE EXPECT; REAL EXPECT; FORWARD;
PRT(430) = POLISHER  PROCEDURE INLINE; FORWARD;
PRT(431) = INLINE    PROCEDURE SUBHAND(FROM); VALUE FROM; BOOLEAN FROM; FORWARD;
PRT(432) = SUBHAND   PROCEDURE IOSTMT; FORWARD;
PRT(433) = IOSTMT    INTEGER PROCEDURE IFEXP; FORWARD;
PRT(434) = IFEXP     PROCEDURE PARSE; FORWARD;
PRT(435) = PARSE     PROCEDURE DOT; FORWARD;
PRT(436) = DOT       PROCEDURE IFCLAUSE; FORWARD;
PRT(437) = IFCLAUSE  INTEGER PROCEDURE GET(SYLLABLE); VALUE SYLLABLE; REAL SYLLABLE; FORWARD;
PRT(440) = GET       INTEGER PROCEDURE GNAT(L); VALUE L; REAL L; FORWARD;
PRT(441) = GNAT      PROCEDURE PANA; FORWARD;

```

```

02985500 T 0205
02986000 T 0205
02986500 T 0205
02987000 T 0205
02987500 T 0205
03001000 T 0205
03002000 T 0205
03003000 T 0205
03004000 T 0205
03005000 T 0205
03006000 T 0205
03007000 T 0205
03009000 T 0205
03010000 T 0205
03011000 T 0205
03012000 T 0205
03013000 T 0205
03014000 T 0205
03015000 T 0205
03018000 T 0205
03019000 T 0205
03020000 T 0205
03021000 T 0205

```

PRT(442) = PANA	PROCEDURE IFSTMT; FORWARD;	03022000 T	0205
PRT(443) = IFSTMT	PROCEDURE GOGEN(LABELBAT, BRANCHTYPE);	03023000 T	0205
PRT(444) = GOGEN	VALUE LABELBAT, BRANCHTYPE;	03024000 T	0205
	REAL LABELBAT, BRANCHTYPE; FORWARD;	03025000 T	0205
	BOOLEAN PROCEDURE SIMPGO; FORWARD;	03026000 T	0205
PRT(445) = SIMPGO	PROCEDURE STMT; FORWARD;	03027000 T	0205
PRT(446) = STMT	PROCEDURE EMIT(SYLLABLE); VALUE SYLLABLE; REAL SYLLABLE; FORWARD;	03028000 T	0205
PRT(447) = EMIT	PROCEDURE PROCSTMT(FROM); VALUE FROM; BOOLEAN FROM; FORWARD;	03029000 T	0205
PRT(450) = PROCSTMT	PROCEDURE STRMPROCSTMT; FORWARD;	03030000 T	0205
PRT(451) = STRMPROCSTMT	PROCEDURE CONSTANTCLEAN; FORWARD;	03034000 T	0205
PRT(452) = CONSTANTCLEAN	PROCEDURE SCATTERELBAT; FORWARD;	03035000 T	0205
PRT(453) = SCATTERELBAT	PROCEDURE EMITB(BRANCH, FROM, TOWARDS); VALUE BRANCH, FROM, TOWARDS;	03036000 T	0205
PRT(454) = EMITB	INTEGER BRANCH, FROM, TOWARDS; FORWARD;	03037000 T	0205
	PROCEDURE VARIABLE(FROM); INTEGER FROM; FORWARD;	03038000 T	0205
PRT(455) = VARIABLE	PROCEDURE IMPFUN; FORWARD;	03039000 T	0205
PRT(456) = IMPFUN	PROCEDURE RIGHT(L); VALUE L; INTEGER L; FORWARD;	03039500 T	0205
PRT(457) = RIGHT	PROCEDURE STREAMSTMT; FORWARD;	03040000 T	0205
PRT(460) = STREAMSTMT	PROCEDURE SEGMENTSTART(B); VALUE B; BOOLEAN B; FORWARD;	03041000 T	0205
PRT(461) = SEGMENTSTART	PROCEDURE SEGMENT(SIZE, FR); VALUE SIZE, FR; INTEGER SIZE, FR; FORWARD;	03042000 T	0205
PRT(462) = SEGMENT		03043000 T	0205
		03044000 T	0205
		03045000 T	0205
		03046000 T	0205
PRT(463) = BAE	INTEGER PROCEDURE BAE; FORWARD;	03047000 T	0205
PRT(464) = PROGDESCBLDR	PROCEDURE PROGDESCBLDR(A, B, C, D); VALUE A, B, C, D;	03047100 T	0205
	INTEGER A, C, D; BOOLEAN B; FORWARD;	03048000 T	0205
PRT(465) = BANA	PROCEDURE BANA; FORWARD;	03049000 T	0205
PRT(466) = EMITNUM	PROCEDURE EMITNUM(A); VALUE A; REAL A; FORWARD;	03050000 T	0205
PRT(467) = EMITD	PROCEDURE EMITD(A, B, T); VALUE A, B, T; INTEGER A, B, T; FORWARD;	03051000 T	0205
PRT(470) = GETSPACE	INTEGER PROCEDURE GETSPACE(S, L); VALUE S, L;	03051001 T	0205
	INTEGER L; BOOLEAN S; FORWARD;	03052000 T	0205
PRT(471) = FORSTMT	PROCEDURE FORSTMT; FORWARD;	03053000 T	0205
	REAL PROCEDURE TAKE(INDEX); VALUE INDEX; INTEGER INDEX; FORWARD;	03054000 T	0205
	PROCEDURE E; FORWARD;		

PRT(472) = E	PROCEDURE ENTRY(TYPE); VALUE TYPE; REAL TYPE; FORWARD;	03055000 T	0205
PRT(473) = ENTRY	PROCEDURE PUTNBUMP(P1); VALUE P1; REAL P1; FORWARD;	03057000 T	0205
PRT(474) = PUTNBUMP	PROCEDURE JUMPCHKNX; FORWARD;	03058000 T	0205
PRT(475) = JUMPCHKNX	PROCEDURE JUMPCHKX; FORWARD;	03059000 T	0205
PRT(476) = JUMPCHKX	PROCEDURE DBLSTMT; FORWARD;	03060000 T	0205
PRT(477) = DBLSTMT	PROCEDURE BLOCK(S); VALUE S; BOOLEAN S; FORWARD;	03067000 T	0205
PRT(500) = BLOCK	PROCEDURE PURGE(STOPPER); VALUE STOPPER; REAL STOPPER; FORWARD;	03068000 T	0205
PRT(501) = PURGE	PROCEDURE ENTER(TYPEV);	03069000 T	0205
PRT(502) = ENTER	VALUE TYPEV;	03070000 T	0205
	REAL TYPEV; FORWARD;	03071000 T	0205
		03072000 T	0205
		03073000 T	0205
		03074000 T	0205
		03075000 T	0205
	COMMENT THIS SECTION CONTAINS THE EMITTERS. THEY ARE THE AGENTS WHICH	04000000 T	0205
	ACTUALLY PRODUCE CODE AND DEBUGGING OUTPUT;	04001000 T	0205
	COMMENT EMITL EMITS A LIT CALL;	04002000 T	0205
	PROCEDURE EMITL(LITERAL); VALUE LITERAL; INTEGER LITERAL;	04003000 T	0205
PRT(503) = EMITL	EMIT(0&LITERAL[36:38:10]);	04004000 T	0205
	COMMENT EMITO EMIT AN OPERATOR;	04005000 T	0207
	PROCEDURE EMITO(OPERATOR); VALUE OPERATOR; INTEGER OPERATOR;	04006000 T	0207
PRT(504) = EMITO	EMIT(1&OPERATOR[36:38:10]);	04007000 T	0207
	COMMENT EMITC IS PRIMARILY FOR USE BY STRMSTMT TO EMIT CHARACTER MODE	04008000 T	0209
	OPERATORS, HOWEVER IT ALSO HANDLES DIA, DIB, AND TRB;	04009000 T	0209
	PROCEDURE EMITC(REPEAT, OPERATOR); VALUE REPEAT, OPERATOR;	04010000 T	0209
PRT(505) = EMITC	INTEGER REPEAT, OPERATOR;	04011000 T	0209
	BEGIN	04012000 T	0209
	IF REPEAT ≥ 264 THEN FLAG(268);	04013000 T	0209
	EMIT(OPERATOR&REPEAT[36:42:6]) END EMITC;	04014000 T	0211
	COMMENT EMITV EMITS AN OPERAND CALL. IF THE ADDRESS IS FOR THE SECOND	04015000 T	0213
	HALF OF THE PRT, THEN IT ALSO EMITS A PRTE;	04016000 T	0213
	PROCEDURE EMITV(ADDRESS); VALUE ADDRESS; INTEGER ADDRESS;	04017000 T	0213
PRT(506) = EMITV			

BEGIN IF ADDRESS > 1023 THEN EMIT0(PRTE);	04018000 T	0213
EMIT(2 & ADDRESS [36:38:10]) END EMITV;	04019000 T	0215
COMMENT EMITN EMITS A DESCRIPTOR CALL. IF THE ADDRESS IS FOR THE	04020000 T	0217
SECOND HALF OF THE PRT, THEN IT ALSO EMITS A PRTE;	04021000 T	0217
PROCEDURE EMITN(ADDRESS); VALUE ADDRESS; INTEGER ADDRESS;	04022000 T	0217
PRT(507) = EMITN	04023000 T	0217
BEGIN IF ADDRESS > 1023 THEN EMIT0(PRTE);	04024000 T	0219
EMIT(3 & ADDRESS [36:38:10]) END EMITN;		
COMMENT EMITPAIR EMITS A LITC ADDRESS FOLLOWED BY OPERATOR. IF THE	04025000 T	0221
ADDRESS IS FOR THE SECOND HALF OF THE PRT, THEN IT ALSO	04026000 T	0221
EMITS PRTE;	04027000 T	0221
PROCEDURE EMITPAIR(ADDRESS, OPERATOR);	04028000 T	0221
PRT(510) = EMITPAIR	04029000 T	0221
VALUE ADDRESS, OPERATOR;	04030000 T	0221
INTEGER ADDRESS, OPERATOR;	04031000 T	0221
BEGIN	04032000 T	0221
EMITL(ADDRESS);	04033000 T	0221
IF ADDRESS > 1023 THEN EMIT0(PRTE);	04034000 T	0223
EMIT0(OPERATOR) END EMITPAIR;		
COMMENT ADJUST ADJUST L TO THE BEGINING OF A WORD AND FILLS IN THE	04080000 T	0224
INTERVENING SPACE WITH NOPS. IT CHECKS STREAMTUG TO DECIDE	04081000 T	0224
WHICH SORT OF NOP TO USE;	04082000 T	0224
PROCEDURE ADJUST;	04083000 T	0224
PRT(511) = ADJUST	04084000 T	0224
BEGIN	04085000 T	0224
WHILE L.[46:2]#0 DO EMIT(45);	04086000 T	0224
END ADJUST;	04087000 T	0228
PROCEDURE EMITLNG;	04098000 T	0228
PRT(512) = EMITLNG	04099000 T	0228
BEGIN LABEL E;	04100000 T	0000
IF NOT LINKTOG THEN GO TO E;	04101000 T	0000
COMMENT GO TO E IF LAST THING IS A LINK;	04102000 T	0000
IF GET(L) # 0 THEN GO TO E;	04103000 T	0002
COMMENT EITHER LAST EXPRESSION WAS CONDITIONAL OR THERE IS NO	04104000 T	0002
LNG OR RELATIONAL OPERATOR;	04105000 T	0002
IF GT1 + GET(L-1) = 77 THEN L + L=1	04106000 T	0005
COMMENT LAST THING WAS AN LNG - SO CANCEL IT;	04107000 T	0005
ELSE IF GT1.[42:6]=21 AND GT1.[37:2]#0 THEN % AHA	04108000 T	0009
COMMENT LAST THING WAS A RELATIONAL;		

START OF SEGMENT ***** 26

```

        BEGIN L←L+1; EMITD(REAL(BOOLEAN(GT1,[36:10])) EQV
        BOOLEAN(IF GT1,[40:2] = 0 THEN 511 ELSE 463)))
COMMENT NEGATE THE RELATIONAL; END ELSE
E:      EMITD(LNG) END EMITLNG;

```

```

04109000 T 0009
04110000 T 0012
04111000 T 0015
04112000 T 0015
26 IS 18 LONG, NEXT SEG 3

```

```

COMMENT EMITB EMITS A BRANCH OPERATOR AND ITS ASSOCIATED NUMBER;
PROCEDURE EMITB(BRANCH, FROM, TOWARDS);
    VALUE BRANCH, FROM, TOWARDS;
    INTEGER BRANCH, FROM, TOWARDS;
    BEGIN
        INTEGER TL;

```

```

04113000 T 0228
04114000 T 0228
04115000 T 0228
04116000 T 0228
04117000 T 0228
04118000 T 0228
START OF SEGMENT ***** 27

```

STACK(F+2) = TL

```

        TL ← L;
        IF TOWARDS > FOULED THEN FOULED ← TOWARDS;
        L ← FROM-2;
        GT1 ← TOWARDS-FROM;
        IF TOWARDS,[46:2] = 0
            THEN BEGIN
                BRANCH ← BRANCH&1[39:47:1];
                GT1 ← TOWARDS DIV 4 - (FROM-1) DIV 4 END;
        EMITNUM(ABS(GT1));
        EMITD(BRANCH&(REAL(GT1≥ 0)+1)[42:46:2]);

        L ← TL
    END EMITB;

```

```

04119000 T 0000
04119500 T 0000
04120000 T 0002
04120100 T 0004
04120200 T 0005
04120300 T 0006
04120400 T 0007
04120500 T 0008
04121000 T 0011
04122000 T 0012
04123000 T 0015
04124000 T 0015
04125000 T 0015
27 IS 19 LONG, NEXT SEG 3

```

```

COMMENT DEBUGWORD FORMATS TWO FIELDS FOR DEBUGGING OUTPUT IN
OCTAL, NAMELY :
    1. 4 CHARACTERS FOR THE L REGISTER,
    2. 16 CHARACTERS FOR THE WORD BEING EMITTED. ;

```

```

STREAM PROCEDURE DEBUGWORD( SEQ, CODE, FEIL); VALUE SEQ, CODE ;
PRT(513) = DEBUGWORD

```

```

    BEGIN
        DI←FEIL; SI← LOC SEQ; SI← SI+4; DS ← 4 CHR;
        DS ← 2 LIT" ";
        SI ← LOC CODE ;
        16( DS ← 3 RESET; 3( IF SB THEN DS←SET ELSE
            DS ← RESET ; SKIP 1 SB));
        29(DS ← 2 LIT" " );
    END ;

```

```

04126000 T 0228
04127000 T 0228
04128000 T 0228
04129000 T 0228
04130000 T 0228

```

```

04131000 T 0228
04132000 T 0229
04133000 T 0230
04134000 T 0230
04135000 T 0230
04136000 T 0232
04137000 T 0233
04138000 T 0234

```

```

COMMENT EMITWORD PLACES THE PARAMETER, "WORD", INTO EDUC, IF
DEBUGGING IS REQUIRED, "L" AND "WORD" ARE OUTPUT ON
THE PRINTER FILE IN OCTAL FORMAT, ;
PROCEDURE EMITWORD (WORD); VALUE WORD; REAL WORD;

```

```

04139000 T 0234
04140000 T 0234
04141000 T 0234
04142000 T 0234

```

PRT(514) = EMITWORD

```
BEGIN
ADJUST;
IF L ≥ 4088 THEN BEGIN ERR(200); L+0; END
ELSE BEGIN
MOVE(1,WORD, CODE(L DIV 4+1));
IF DEBUGTOG THEN
BEGIN DEBUGWORD(B2D(L),WORD,LIN);
WRITELINE END;
FOULED + L + L+4; END
END EMITWORD;
```

04143000	T	0234
04144000	T	0234
04145000	T	0235
04146000	T	0238
04147000	T	0240
04148000	T	0245
04149000	T	0245
04150000	T	0248
04151000	T	0258
04152000	T	0260

COMMENT CONSTANTCLEAN IS CALLED AFTER AN UNCONDITIONAL BRANCH HAS BEEN EMITTED. IF ANY CONSTANTS HAVE BEEN ACCUMULATED BY EMITNUM IN INFO(0,*), CONSTANTCLEAN WILL FIX THE CHAIN OF C-RELATIVE OPDC S LEFT BY EMITNUM. IF C-RELATIVE ADDRESSING IS IMPOSSIBLE (I.E. THE ADDRESS IF GREATER THAN 127 WORDS) THEN THE CONSTANT ALONG WITH THE 1ST LINK OF THE OPDC CHAIN IS ENTERED IN INFO. AT PURGE TIME THE REMAINING OPDC S ARE EMITTED WITH F-RELATIVE ADDRESSING AND CODE EMITTED TO STORE THE CONSTANTS INTO THE PROPER F-RELATIVE CELLS. ;

```
PROCEDURE CONSTANTCLEAN ;
IF MRCLEAN THEN
BEGIN
INTEGER J,TEMPL,D,LINK;
```

04153000	T	0260
04154000	T	0260
04155000	T	0260
04156000	T	0260
04157000	T	0260
04158000	T	0260
04159000	T	0260
04160000	T	0260
04161000	T	0260
04162000	T	0260
04163000	T	0260
04164000	T	0260
04165000	T	0261
04166000	T	0261

PRT(515) = *SEGMENT DESCRIPTOR*

```
STACK(F+2) = J
STACK(F+3) = TEMPL
STACK(F+4) = D
STACK(F+5) = LINK
```

```
STACK(F+6) = CREL
```

```
BOOLEAN CREL;
```

```
LABEL ALLTHU ;
```

```
FOR J + 1 STEP 2 UNTIL LASTENTRY DO
```

```
BEGIN
```

```
ADJUST; TEMPL+L; L+INFO(0,255-J+1);
```

```
CREL + FALSE;
```

```
DO BEGIN
```

```
IF D+(TEMPL+L+3)DIV 42128 THEN
```

```
IF MODE ≠ 0 THEN
```

```
BEGIN FLAG(50); GO TO ALLTHU END;
```

```
LINK+GET(L);
```

```
CREL + TRUE;
```

```
IF MODE ≠ 0 THEN EMITV(D+768) ELSE
```

```
EMITV(REAL(TEMPL+2048)*1024+TEMPL DIV 4);
```

```
END UNTIL L+ LINK = 4095 ;
```

START OF SEGMENT ***** 28

04167000	T	0000
04168000	T	0000
04169000	T	0000
04170000	T	0000
04171000	T	0001
04172000	T	0001
04173000	T	0005
04174000	T	0005
04175000	T	0006
04175500	T	0008
04176000	T	0010
04177000	T	0011
04178000	T	0011
04179000	T	0011
04180000	T	0011
04181000	T	0011
04182000	T	0011
04183000	T	0013
04184000	T	0013
04184500	T	0016
04185000	T	0019

```

ALLTHU:      L ← TEMPL;
              IF CREL THEN EMITWORD( INFO[0,255-J ]);
              END;
              LASTENTRY ← 0;
            END ;
PRT(516) = *SEGMENT DESCRIPTOR*

```

```

04186000 T 0021
04187000 T 0022
04188000 T 0025
04189000 T 0028
04190000 T 0028

```

28 IS 33 LONG, NEXT SEG 3

```

COMMENT      EMITNUM HANDLES THE EMISSION OF CODE FOR CONSTANTS,BOTH
              EXPLICIT AND IMPLICIT. IN EVERY CASE,EMITNUM WILL
              PRODUCE CODE TO GET THE DESIRED CONSTANT ON TOP OF
              THE STACK. IF THE NUMBER IS A LITERAL A SIMPLE LITC
              SYLLABLE IS PRODUCED, HOWEVER, NON-LITERALS ARE KEPT
              IN THE ZERO-TH ROW OF INFO WITH THE SYLLABLE
              POSITION,L. THE FIRST EMITNUM ON A PARTICULAR
              CONSTANT CAUSES THE VALUES OF L AND THE CONSTANT
              TO BE STORED IN INFO[0,+] (NOTE: ITEMS ARE STORED
              IN REVERSE STARTING WITH INFO[0,255],ETC.), THEN
              ITS THE JOB OF CONSTANTCLEAN TO EMIT THE ACTUAL
              OPDC (SEE CONSTANTCLEAN PROCEDURE FOR DETAILS) ;
PROCEDURE EMITNUM( C ); VALUE C; REAL C;
              BEGIN LABEL FINISHED,FOUND ; REAL N;

```

```

04191000 T 0266
04192000 T 0266
04193000 T 0266
04194000 T 0266
04195000 T 0266
04196000 T 0266
04197000 T 0266
04198000 T 0266
04199000 T 0266
04200000 T 0266
04201000 T 0266
04202000 T 0266
04203000 T 0266
04204000 T 0266

```

START OF SEGMENT ***** 29

STACK(F+2) = N

```

              IF C,[1:37]=0 THEN EMITL(C)
              ELSE
                BEGIN
                  FOULED ← L;
                  FOR N ← 1 STEP 2 UNTIL LASTENTRY DO
                    IF INFO[0,255-N] = C THEN GO TO FOUND ;
                    INFO[0,255 -LASTENTRY] ← L;
                    INFO[0,255 -LASTENTRY-1] ← C ;
                    EMITN(1023);
                    IF MODE=0 THEN EMITO(NOP);
                    LINKTOG←FALSE;
                    IF LASTENTRY ← LASTENTRY+2 ≥ 128 THEN
                      BEGIN
                        C ← BUMPL;
                        CONSTANTCLEAN;
                        EMITB(BFW,C,L);
                      END;
                    GO TO FINISHED;
                FOUND:  EMIT(INFO[0,255 -N+1]);
                      LINKTOG←FALSE;
                      INFO[0,255-N+1] ← L-1;
                      IF MODE=0 THEN EMITO(NOP);
                      END;
                FINISHED:END EMITNUM ;

```

```

04205000 T 0000
04206000 T 0002
04207000 T 0003
04207500 T 0003
04208000 T 0004
04209000 T 0005
04210000 T 0010
04211000 T 0012
04212000 T 0015
04212100 T 0016
04213000 T 0018
04214000 T 0019
04215000 T 0020
04216000 T 0021
04217000 T 0023
04218000 T 0023
04219000 T 0024
04220000 T 0024
04221000 T 0025
04222000 T 0028
04223000 T 0029
04223100 T 0033
04224000 T 0035
04225000 T 0035

```

29 IS 38 LONG, NEXT SEG 3

COMMENT SEARCH PERFORMS A BINARY SEARCH ON THE COP AND WOP

04226000 T 0266

ARRAYS, GIVEN THE OPERATOR BITS SEARCH YIELDS THE BCD
MNEUMONIC FOR THAT OPERATOR. IF THE OPERATOR CANNOT
BE FOUND SEARCH YIELDS BLANKS.
NOTE: DIA, DIB, TRB ARE RETURNED AS BLANKS. ;
ALPHA PROCEDURE SEARCH (Q, KEY); VALUE KEY; ARRAY Q[0]; REAL KEY ;
PRT(517) = SEARCH

BEGIN LABEL L;
COMMENT GT1 AND GT2 ARE INITIALIZED ASSUMING THAT Q IS ORDERED
BY PAIRS (ARGUMENT, FUNCTION, ARGUMENT, FUNCTION, ETC.)
AND THAT THE FIRST ARGUMENT IS IN Q[4]. FURTHERMORE
THE LENGTH OF Q IS 128, ;
INTEGER N, I ;

STACK(F+3) = N
STACK(F+4) = I

N = 64 ;
FOR I = 66 STEP 1 IF Q[I] < KEY THEN N ELSE = N
WHILE N + N DIV 2 ≥ 1 DO
IF Q[I] = KEY THEN GO TO L ;
I = I + 1 ; COMMENT ARGUMENT NOT FOUND, SEARCH = Q[1] ;
L: SEARCH = Q[I + 1] ;
END SEARCH ;

COMMENT B2D CONVERTS THE FOUR LOW ORDER OCTAL DIGITS TO BCD
CODE ;
ALPHA PROCEDURE B2D(B); VALUE B; REAL B;
B2D = 0 & B[45:45:3] & B[39:42:3] & B[33:39:3] & B[27:36:3] ;

COMMENT PACK IS A STREAM PROCEDURE WHICH INSERTS THE SYLLABLE
INTO THE EDOC ARRAY. THE SPECIFIC ELEMENT OF EDOC
IS PRECISILY = EDOC[(L DIV 4) DIV 128, (< DIV 4) MOD 128]
SYLLABLE POSITION = (L MOD 4), WHERE L IS THE SYLLABLE
NUMBER RELATIVE TO THE BEGINNING OF THE SEGMENT;
STREAM PROCEDURE PACK(WORD, POSITION, SYLLABLE);

PRT(520) = PACK

VALUE POSITION, SYLLABLE;
BEGIN
DI = WORD ; DI = DI + POSITION ; DI = DI + POSITION;
SI = LOG SYLLABLE ; SI = SI + 6;
DS = 2 CHR ;
END PACK ;

COMMENT DEBUG PRINTS OUT OBJECT CODE IF "DEBUGN" IS SET;
PROCEDURE DEBUG(S); VALUE S; REAL S;
PRT(521) = DEBUG

BEGIN REAL T1;

04227000 T 0266
04228000 T 0266
04229000 T 0266
04230000 T 0266
04231000 T 0266

04232000 T 0266
START OF SEGMENT ***** 30

04233000 T 0000
04234000 T 0000
04235000 T 0000
04236000 T 0000
04237000 T 0000

04238000 T 0000
04239000 T 0000
04240000 T 0005
04241000 T 0009
04242000 T 0011
04243000 T 0011
04244000 T 0013

30 IS 16 LONG, NEXT SEG 3

04245000 T 0266
04246000 T 0266
04247000 T 0266
04248000 T 0266

04265000 T 0274
04266000 T 0274
04267000 T 0274
04268000 T 0274
04269000 T 0274
04270000 T 0274

04271000 T 0274
04272000 T 0274
04273000 T 0275
04274000 T 0276
04275000 T 0276
04276000 T 0277

04277000 T 0277
04277500 T 0277

04278000 T 0277
START OF SEGMENT ***** 31

STACK(F+2) = T1		04278500 T 0000	
IF SINGLTOG THEN		04279000 T 0000	
WRITE(LINE,BUG,B2D(L),			
PRT(522) = *LIST, LABEL, OR SEGMENT DESCRIPTOR*		04279500 T 0008	
IF STREAMTOG THEN		04280000 T 0009	
SEARCH(COP,S,[42:6])		04280500 T 0010	
ELSE IF T1=S,[46:2]=1 THEN SEARCH(WOP,S,[36:10])		04281000 T 0015	
ELSE WOP[T1], IF STREAMTOG THEN		04281500 T 0019	
B2D(S,[36:6]) ELSE IF T1=1 THEN WOP[1]		04282000 T 0023	
ELSE B2D(S,[36:10]),B2D(S))		04282500 T 0027	
ELSE WRITE(LINE,BUG,B2D(L),			
PRT(523) = *LIST, LABEL, OR SEGMENT DESCRIPTOR*		04283000 T 0038	
IF STREAMTOG THEN		04283500 T 0039	
SEARCH(COP,S,[42:6])		04284000 T 0040	
ELSE IF T1=S,[46:2]=1 THEN SEARCH(WOP,S,[36:10])		04284500 T 0045	
ELSE WOP[T1], IF STREAMTOG THEN		04285000 T 0049	
B2D(S,[36:6]) ELSE IF T1=1 THEN WOP[1]		04285500 T 0053	
ELSE B2D(S,[36:10]),B2D(S))		04286000 T 0061	
END DEBUG;	31 IS 64 LONG, NEXT SEG 3		
COMMENT EMIT PLACES SYLLABLES INTO EDOC, CALLS DEBUG FOR		04288000 T 0277	
DEBUGGING OUTPUT ON THE PRINTER, AND CHECKS FOR SEGMENTS		04289000 T 0277	
GREATER THAN 4093 SYLLABLES,		04290000 T 0277	
PROCEDURE EMIT (S) ; VALUE S; REAL S ;		04291000 T 0277	
BEGIN		04292000 T 0277	
IF L < 4088 THEN		04293000 T 0277	
BEGIN		04294000 T 0278	
LINKTOG + TRUE;		04295000 T 0279	
PACK(CODE(L DIV 4+1),L,[46:2],S);		04296000 T 0280	
IF DEBUGTOG THEN DEBUG(S);		04297000 T 0285	
L=L+1;		04298000 T 0287	
END ELSE		04299000 T 0288	
BEGIN ERR(200); L+1; END;		04300000 T 0288	
COMMENT 200 EMIT = SEGMENT GREATER THAN 4093 SYLLABLES	*	04301000 T 0292	
END EMIT ;		04302000 T 0292	
COMMENT EMITD EMITS THE DIA,DIB,TRB SEQUENCE OF CODE. THE		04305000 T 0292	
PREVIOUS SETTING OF THE G=H AND K=V REGISTERS IS COMPARED		04306000 T 0292	
THE CURRENT , IF THE G=H,K=V OR BOTH ARE ALREADY SET THEN		04307000 T 0292	
THE APPROPRIATE SYLLABLE(S) ARE OMITTED		04308000 T 0292	
IF 0 BITS ARE TO BE TRANSFERED THEN NO SYLLABLES ARE		04309000 T 0292	
EMITTED ;		04310000 T 0292	
PROCEDURE EMITD(A,B,T); VALUE A,B,T ; INTEGER A,B,T;		04311000 T 0292	
BEGIN LABEL EXIT,NORMAL;		04311010 T 0292	
REAL Q;	START OF SEGMENT ***** 32		
STACK(F+2) = Q		04311020 T 0000	
IF T = 15 THEN		04311030 T 0000	
BEGIN		04311040 T 0000	
IF A = 33 THEN Q + 512		04311050 T 0001	

```

ELSE IF A # 18 THEN GO TO NORMAL;
IF B = 18 THEN Q = Q+256
ELSE IF B # 33 THEN GO TO NORMAL;
EMIT(Q+197); COMMENT -- THIS GETS OUT FIXED FIELD;
GO TO EXIT;
END;

NORMAL:
IF T# 0 THEN
BEGIN

    EMIT(((DIALA+A) DIV 6)*512 + ( A+ A MOD 6)* 64 + DIA);

    EMIT(((DIALB+B) DIV 6)*512 + ( B+ B MOD 6)* 64 + DIB);
    EMIT(TRB+64*T);

END

EXIT:      END      EMITD  ;
END;

```

```

04311060 T 0002
04311070 T 0005
04311080 T 0006
04311090 T 0009
04311100 T 0010
04311110 T 0011
04311120 T 0011
04312000 T 0011
04313000 T 0011
04314000 T 0012
04315000 T 0012
04316000 T 0017
04317000 T 0017
04318000 T 0021
04319000 T 0023
04320000 T 0023
04321000 T 0023
04322000 T 0023
04322100 T 0023
32 IS 27 LONG, NEXT SEG 3

```

```

PROCEDURE EMIT1(E,A,B); VALUE E,A,B; REAL E,A,B;
PRT(524) = EMIT1
    BEGIN LABEL EXIT,IS;
        INTEGER S,T1,T2;

        STACK(F+2) = S
        STACK(F+3) = T1
        STACK(F+4) = T2

        PRT(525) = EMIT21

```

```

04500000 T 0292
04501000 T 0292
START OF SEGMENT ***** 33
04502000 T 0000

```

```

PROCEDURE EMIT21(E,B); VALUE E,B;
    REAL E;
    BOOLEAN B;
    BEGIN IF E = 0 THEN
        BEGIN IF B THEN EMITD(XCH); END
    ELSE BEGIN GT1 = E.ADDRESS;
        IF E + E.CLASS ≤ INTID THEN
            EMITV(GT1)
        ELSE IF E ≤ INTARRAYID THEN
            EMITPAIR(GT1,LOD) ELSE
            EMITN(GT1)
        END
    END;
END;

```

```

04503000 T 0000
04504000 T 0000
04505000 T 0000
04506000 T 0000
04507000 T 0000
04508000 T 0002
04509000 T 0004
04521000 T 0006
04522000 T 0007
04523000 T 0008
04524000 T 0010
04525000 T 0011
04526000 T 0011

```

```

IF B = 0 THEN
    BEGIN EMIT21(E,FALSE); GO TO EXIT END;
IF STACKCT # 0 THEN GO TO IS;
IF B = 15 THEN
    BEGIN IF A = 33 THEN
        BEGIN EMIT21(E,FALSE);
            EMIT(0); EMITD(INX);

```

```

04526100 T 0011
04526200 T 0012
04527000 T 0014
04528000 T 0016
04529000 T 0016
04530000 T 0018
04531000 T 0019

```

```

                                GO TO EXIT;
                                END;
                                IF A = 18 THEN
                                    BEGIN EMIT(0);
                                        EMIT21(E,TRUE);
                                        EMIT(197);
                                        GO TO EXIT;
                                    END;
                                GO TO IS;
                                END;
                                IF B ≤ 10 AND A+B = 48 THEN
                                    BEGIN EMIT21(E,FALSE);
                                        EMITL(2*B-1);
                                    END;
                                EMIT(LND);
                                GO TO EXIT;
                                END;
IS:                                IF (S + (48-A-B) MOD 6)+B ≤ 39 THEN
                                    BEGIN EMIT21(E,FALSE);
                                        EMIT(T2+(T1+A DIV 6)*512+(A MOD 6)*64+DIA);
                                        EMIT(((A+B-1) DIV 6 -T1+1)*512+64*S+37);
                                        GO TO EXIT;
                                    END;
                                EMIT(0);
                                EMIT21(E,TRUE);
                                EMITD(A,48-B,B);
                                EXIT: END;

```

PRT(526) = LN
PRT(527) = EXP
PRT(530) = X TO THE I

PRT(531) = STEPIT

PRT(532) = STEP1
INTEGER PROCEDURE STEP1; STEP1←ELCLASS+TABLE(I+I+1);

COMMENT TAKE FETCHS A WORD FROM INFO;
REAL PROCEDURE TAKE(INDEX); VALUE INDEX; INTEGER INDEX;
TAKE ← INFO[INDEX,LINKR,INDEX,LINKC];

COMMENT PUT PLACES A WORD INTO INFO;

04532000 T 0021
04533000 T 0021
04534000 T 0021
04535000 T 0022
04536000 T 0023
04537000 T 0024
04538000 T 0025
04539000 T 0025
04540000 T 0025
04541000 T 0026
04542000 T 0026
04543000 T 0028
04544000 T 0030

04545000 T 0033
04546000 T 0034
04547000 T 0034
04548000 T 0034
04549000 T 0038
04550000 T 0039
04551000 T 0044
04552000 T 0049
04553000 T 0050
04554000 T 0050
04555000 T 0051
04556000 T 0052
04557000 T 0053
04558000 T 0053

33 IS 37 LONG, NEXT SEG 3

05000000 T 0292
05001000 T 0292
05002000 T 0292

05003000 T 0295

05004000 T 0301
05005000 T 0301
05006000 T 0301

05007000 T 0307

PRT(533) = PUT	PROCEDURE PUT(WORD,INDEX); VALUE WORD,INDEX; REAL WORD,INDEX;	05008000 T	0307
	INFO[INDEX,LINKR,INDEX,LINKC] + WORD;	05009000 T	0307
	COMMENT FLAG FLAGS ERROR MESSAGES, COUNTS THEM AND SUPPRESS FUTURE ERROR MESSAGES UNTIL THE COMPILER THINKS IT HAS RECOVERED;	05010000 T	0311
	PROCEDURE FLAG(ERRNUM); VALUE ERRNUM; INTEGER ERRNUM;	05011000 T	0311
	BEGIN	05012000 T	0311
	COMMENT WRITERERROR IS THE STREAM PROCEDURE WHICH ACTUALLY PRODUCES THE ERROR MESSAGE ON THE PRINTER;	05013000 T	0311
	STREAM PROCEDURE WRITERERROR(ERRNUM,ACCUM,LINE,COUNT,LSTSEQ);	05014000 T	0311
		05015000 T	0311
		05016000 T	0311
PRT(534) = WRITERERROR	START OF SEGMENT ***** 34		
	VALUE ERRNUM,COUNT;	05017000 T	0000
	BEGIN	05018000 T	0000
	DI + LINE; 44(DS+2LIT" "); COMMENT CLEAR BUFFER;	05019000 T	0000
	SI + LSTSEQ; SI + SI-8; DS + WDS;	05020000 T	0001
	SI + LINE; DS + 2 WDS;	05021000 T	0002
	4(DS + 2 LIT "XX"); COMMENT SET RIGHT MARGIN FLAG;	05022000 T	0002
	SI + LSTSEQ; DI + LSTSEQ; DI + DI-8; DS + WDS;	05023000 T	0003
	DI + LINE; DI + DI+8; COMMENT INDENT MESSAGE;	05024000 T	0004
	DS + 13 LIT "ERROR NUMBER ";	05025000 T	0005
	SI + LOC ERRNUM; DS + 3 DEC; COMMENT CONVERT ERRNUM;	05026000 T	0007
	DS + 4 LIT " = ";	05027000 T	0007
	SI + ACCUM; SI + SI+3; DS + COUNT CHR;	05028000 T	0008
	COMMENT PLACE ALPHA IN BUFFER;	05029000 T	0009
	DS + LIT " , ";	05030000 T	0009
	END WRITERERROR;	05031000 T	0009
	IF ERRORTOG THEN % DO NOTHING IF WE SUPPRESS MSSGS.	05032000 T	0010
	BEGIN	05033000 T	0010
	SPECTOG := FALSE;	05034000 T	0010
	ERRORCOUNT := ERRORCOUNT+1; COMMENT COUNT ERRORS;	05035000 T	0011
	IF NOT LISTER THEN	05036000 T	0012
	BEGIN	05037000 T	0013
	EDITLINE(LIN,FCR,L DIV 4,L.[46:2],MEDIUM,0);	05038000 T	0013
	MOVE(1,INFO[LASTSEQROW,LASTSEQUENCE],LIN[12]);	05039500 T	0017
	IF NOHEADING THEN DATIME; WRITELINE;	05039600 T	0019
	END;	05041000 T	0031
	COMMENT PRINT CARDIMAGE IF WE ARE NOT LISTING;	05042000 T	0031
	ACCUM[1] + Q; COMMENT RESTORE ACCUMULATOR;	05043000 T	0031
	WRITERERROR(ERRNUM,ACCUM[1],LIN[*],Q,[12:6],	05044000 T	0032
	INFO[LASTSEQROW,LASTSEQUENCE]);	05045000 T	0034
	IF NOT NOHEADING THEN WRITELINE;	05046000 T	0036
	ERRORTOG + FALSE; COMMENT INHIBIT MESSAGES;	05047000 T	0047
	IF PUNCHTOG THEN	05048000 T	0048
	BEGIN REAL T1,T2,T3,T4; LABEL L,L1,EXIT;	05049000 T	0048
PRT(535) = *SEGMENT DESCRIPTOR*	START OF SEGMENT ***** 35		
STACK(F+2) = T1			
STACK(F+3) = T2			

STACK(F+4) = T3
STACK(F+5) = T4

PRT(536) = P1

```
STREAM PROCEDURE P1(L,P); VALUE P;
    BEGIN DI + L; 15(DS+8LIT" ");
    SI + LOC P; DI+L; SI+SI+5; SKIP 3 SB;
    5
    (DS+3 RESET;3(IF SB THEN DS+SET ELSE DS+RESET;SKIP 1 SB));
    SI+P;
    DI+DI+2; 2(8
    (DS+3 RESET;3(IF SB THEN DS+SET ELSE DS+RESET;SKIP 1 SB));
    DS+LIT" ") END;
```

05050000 T 0000
05051000 T 0000
05052000 T 0002
05053000 T 0003
05054000 T 0003
05055000 T 0005
05056000 T 0006
05057000 T 0006
05058000 T 0009

PRT(537) = P2

```
STREAM PROCEDURE P2(L,P); VALUE P;
    BEGIN DI+L; DI +DI+26;DS+LIT"0";
    SI+P; SI+SI-2; SKIP 1SB;
    3
    (DS+3 RESET;3(IF SB THEN DS+SET ELSE DS+RESET;SKIP 1 SB));
    END;
```

05059000 T 0010
05060000 T 0010
05061000 T 0012
05062000 T 0012
05063000 T 0012
05064000 T 0015

PRT(540) = ABS

```
REAL STREAM PROCEDURE ABS(A); VALUE A;
    BEGIN DI + LOC ABS; SI +A; DS+ WDS;DI+DI+8; DS+RESET END;
```

05065000 T 0015
05066000 T 0015

PRT(541) = BITEDUST

```
STREAM PROCEDURE BITEDUST(X,N,ID,LINE,COUNT); VALUE ID,N,COUNT;
    BEGIN LOCAL T,F,H;
    DI+LOC F; SI+LINE; DS+WDS;DI+F;
    SI+LOC ID;SI+SI+2;DS+ 6CHR; 57(DS+2 LIT" ");
    SI + LOC COUNT ;SI+SI+8 ;
    DI +LOC H ; DS +WDS ;
    DI + LOC LINE ; SI + H;
    SI+LOC X;SKIP 2 SB;
    IF SB THEN
    BEGIN SI+X; T+SI;
    N(
    DI+LOC F; SI+LINE; DS+WDS;DI+F;
    SI+ LOC COUNT; SI+SI+6;
    4
    (DS+3 RESET;3(IF SB THEN DS+SET ELSE DS+RESET;SKIP 1 SB));
    DS+2 LIT" "; SI+COUNT; SI+SI+48;COUNT+SI;
    SI+T;
    6(2(8
    (DS+3 RESET;3(IF SB THEN DS+SET ELSE DS+RESET;SKIP 1 SB));
    DS+ LIT" "); DS+LIT" " );
    DI + LOC LINE ; SI + H;
    SI+T;SI+SI+48;T+SI);END END;
```

05067000 T 0018
05068000 T 0018
05069000 T 0019
05070000 T 0020
05071000 T 0021
05071100 T 0022
05071200 T 0022
05072000 T 0023
05073000 T 0023
05074000 T 0024
05075000 T 0025
05076000 T 0026
05077000 T 0027
05078000 T 0027
05079000 T 0027
05080000 T 0030
05081000 T 0031
05082000 T 0032
05083000 T 0032
05084000 T 0035
05085000 T 0036
05086000 T 0037

```

        BITEDUST(ERRORCOUNT, 63, "PRT ", LINE, 21);
FOR T1← 0 STEP 1 UNTIL 3100 BITEDUST(INFO[ T1,*],43,"INFO ",LINE,0);
BITEDUST(      ELBAT[*],13,"ELBAT ",LINE,0);
        BITEDUST(STACKHEAD[*],21,"STHEAD",LINE,0);
        T1←MKABS(ERRNUM)+1; T3←T1;
L1:    T2←ABS(T3);
        IF T2.[33:15]≠0 THEN BEGIN T3←T2.[18:15];
            IF T3≠0 THEN GO TO EXIT ELSE GO TO L END ;
        T4 ← IF T2.[33:15] < 512 THEN 0 ELSE T2.[33:15]&T2[40:10:2];
        L1:P1(LIN[0],T1);IF T1≠T3 THEN BEGIN IF T4≠0 THEN P2(LIN[0],T4);
            T1←T1+1;
            WRITELINE;
                T3←T2.[18:15];GO TO L END;
            T1:=T1-1; WRITELINE;
            GO TO L1;
        EXIT; END ;
PRT(542) = *SEGMENT DESCRIPTOR*

```

END END FLAG;

```

        LABEL ENDOFITALL;
COMMENT ERR# IS THE SAME AS FLAG EXCEPT THAT IT MAKES AN ATTEMPT TO
        RECOVER FROM ERROR SITUATIONS BY SEARCHING FOR A
        SEMICOLON, END, OR BEGIN;
PROCEDURE ERR(ERRNUM); VALUE ERNUM; INTEGER ERRNUM;
        BEGIN FLAG(ERRNUM);
            I ← I+1;
            IF ERNUM = 200 THEN I:=I/0;% SEGMENT TOO LARGE
            IF ERNUM = 611 THEN I:=I/0;% ERRMAX EXCEEDED
            DO IF STEPI = BEGINV THEN STMT UNTIL
                ELCLASS = ENDV OR ELCLASS = SEMICOLON END ERR;

```

```

        DEFINE ERROR = ERR#; COMMENT ERROR IS A SYNONYM FOR ERR;
COMMENT CHECKER IS A SMALL PROCEDURE THAT CHECKS TO SEE THAT THE
        UPLEVEL ADDRESSING CONVENTIONS ARE OBEYED;
PROCEDURE CHECKER(ELBATWORD); VALUE ELBATWORD; REAL ELBATWORD;

```

PRT(543) = CHECKER

```

        BEGIN
            IF MODE ≥ 2 THEN
                IF GT1 ← ELBATWORD.LVL ≥ FRSTLEVEL THEN
                    IF GT1 < SUBLEVEL THEN
                        IF ELBATWORD.[9:2] ≠ 1
                            THEN BEGIN FLAG(101); ERRORTOG ← TRUE END
            END CHECKER;

```

```

05087000 T 0038
05088000 T 0041
05089000 T 0049
05090000 T 0052
05091000 T 0055
05092000 T 0058
05093000 T 0060
05094000 T 0063
05095000 T 0065
05096000 T 0070
05097000 T 0076
05097500 T 0077
05098000 T 0091
05099000 T 0093
05099100 T 0104
05100000 T 0105

```

```

35 IS 106 LONG, NEXT SEG 34
05101000 T 0050
34 IS 93 LONG, NEXT SEG 3

```

```

05101100 T 0311
05102000 T 0311
05103000 T 0311
05104000 T 0311
05105000 T 0311
05106000 T 0311
05107000 T 0312
05107100 P 0314
05107200 P 0316
05108000 T 0319
05109000 T 0321

```

```

05110000 T 0323
05111000 T 0323
05112000 T 0323
05113000 T 0323

```

```

05114000 T 0323
05115000 T 0323
05116000 T 0324
05117000 T 0327
05118000 T 0328
05119000 T 0329
05120000 T 0332

```

COMMENT GIT IS USED TO OBTAIN THE INDEX TO ADDITIONAL INFORMATION
 GIVEN THE LINK TO THE ELBAT WORD;
 INTEGER PROCEDURE GIT(L); VALUE L; REAL L;
 GIT ← TAKE(L).INCR+L.LINK;

05121000 T 0332
 05122000 T 0332
 05123000 T 0332
 05124000 T 0332

COMMENT GNAT IS USED TO OBTAIN THE PRT ADDRESS OF A GIVEN DESCRIPTOR.
 IF THE ADDRESS HAS NOT BEEN ASSIGNED, THEN IT USES
 GETSPACE TO OBTAIN THE PRT ADDRESS;
 INTEGER PROCEDURE GNAT(L); VALUE L; REAL L;
 BEGIN
 REAL A;

05125000 T 0338
 05126000 T 0338
 05127000 T 0338
 05128000 T 0338
 05129000 T 0338
 05130000 T 0338

START OF SEGMENT ***** 36

STACK(F+3) = A

IF GNAT ← (A+TAKE(L)).ADDRESS=0
 THEN PUT(A&(GNAT+GETSPACE(TRUE,L.LINK+1))[16:37:11],L)
 END GNAT;

05131000 T 0000
 05132000 T 0002
 05133000 T 0007
 36 IS 10 LONG, NEXT SEG 3

REAL PROCEDURE TAKEFRST;
 PRT(544) = TAKEFRST

TAKEFRST ← TAKE(ELBAT[I],LINK+ELBAT[I].INCR);

05188000 T 0338
 05189000 T 0338

COMMENT STUFFF DIALS THE F-REGISTER INTO THE F-REGISTER FIELD OF A
 DESCRIPTOR. THE DESCRIPTOR REMAINS ON THE TOP OF THE
 STACK;

PROCEDURE STUFFF(ADDRESS); VALUE ADDRESS; INTEGER ADDRESS;
 PRT(545) = STUFFF

BEGIN
 EMITPAIR(ADDRESS,LOD);
 EMITN(512);
 EMITD(33,18,15) END STUFFF;

05196000 T 0344
 05197000 T 0344
 05198000 T 0344
 05199000 T 0344
 05200000 T 0344
 05201000 T 0344
 05202000 T 0346
 05203000 T 0346

COMMENT LOCAL IS USED TO SEE WHETHER OR NOT A LABEL IS LOCAL TO OUR
 PRESENT CODE;

BOOLEAN PROCEDURE LOCAL(ELBATWORD);
 PRT(546) = LOCAL

VALUE ELBATWORD; REAL ELBATWORD;
 BEGIN IF ELBATWORD.LVL = LEVEL AND
 NOT BOOLEAN(ELBATWORD.FORMAL) THEN
 LOCAL ← TRUE END LOCAL;

05204000 T 0348
 05205000 T 0348
 05206000 T 0348

05207000 T 0348
 05208000 T 0348
 05209000 T 0350
 05210000 T 0351

COMMENT PASSFORMAT COMPILES CODE THAT PASSES A FORMAT. TWO ITEMS ARE

05211000 T 0355


```

        PASSED = THE ARRAY REFERENCING FORMAT TABLE AND THE
        STARTING INDEX. THE ROUTINE HANDLES SUPERFORMATS ALSO;
    PROCEDURE PASSFORMAT;
PRT(547) = PASSFORMAT
        BEGIN INTEGER ADRES;

        CHECKER(ELBAT[I]);
        ADRES = ELBAT[I].ADDRESS;
        IF BOOLEAN(ELBAT[I].FORMAL)
        THEN BEGIN EMITV(ADRES); ADRES = ADRES+1 END
        ELSE BEGIN
            IF TABLE(I) = SUPERFRMTID
            THEN EMITL(TAKEFRST) ELSE EMITL(ELBAT[I].INCR)
        END;
        IF TABLE(I) = SUPERFRMTID
        THEN BEGIN BANAS I = I+1;
            EMITC(SSP); EMITC(ADD); EMITV(ADRES) END;
        EMITPAIR(ADRES,LOD) END PASSFORMAT;

```

```

05212000 T 0355
05213000 T 0355
05214000 T 0355

05215000 T 0355
START OF SEGMENT ***** 37

```

```

05216000 T 0000
05217000 T 0001
05218000 T 0002
05219000 T 0002
05220000 T 0006
05221000 T 0006
05222000 T 0007
05223000 T 0010
05224000 T 0011
05225000 T 0012
05226000 T 0014
05227000 T 0017
37 IS 21 LONG, NEXT SEG 3

```

```

    COMMENT STREAMWORDS EITHER RESERVES OR UNRESERVES STREAM RESERVED
    WORDS = IT COMPLEMENTS THEIR STATE;
    PROCEDURE STREAMWORDS;
PRT(550) = STREAMWORDS
        BEGIN GT1 = 0;
        DO BEGIN
            INFO[1,GT1].LINK=STACKHEAD[GT2+(T+INFO[1,GT1]).ADDRESS];
            STACKHEAD[GT2] = T.LINK;
            GT1 = GT1+2;
        END UNTIL BOOLEAN(T.FORMAL) END STREAMWORDS;

```

```

05228000 T 0355
05229000 T 0355
05230000 T 0355

05231000 T 0355
05232000 T 0356
05233000 T 0357
05234000 T 0361
05235000 T 0364
05236000 T 0366

```

```

PROCEDURE PROGDESCBLDR(PRTADR,SAV,SIZE,TYPE);
        VALUE PRTADR,SAV,SIZE,TYPE;
        INTEGER PRTADR,SIZE,TYPE; BOOLEAN SAV;
        BEGIN PRTADR=PRTADR,[38:10];
        IF SAV THEN BEGIN PRT[PRTADR] = ( IF TYPE = LDES
        THEN SIZE ELSE CORADR)
            &SIZE[8:38:10]&TYPE[1:43:15]&3[6:46:2];
            IF TYPE=LDES THEN CORADR=CORADR+SIZE;
        END
        ELSE BEGIN PRT[PRTADR]=0&DISKADR[18:33:15]&SIZE[8:38:10]
            &TYPE[1:43:15]&1[6:46:2];

```

```

05237000 T 0367
05238000 T 0367
05239000 T 0367
05240000 T 0367
05241000 T 0367
05242000 T 0367
05243000 T 0367
05244000 T 0367
05245000 T 0367
05246000 T 0367
05247000 T 0367
05247500 T 0367
05248000 T 0369
05248500 T 0371
05249000 T 0373
05250000 T 0376
05251000 T 0379
05252000 T 0379
05253000 T 0382

```

```

        DISKADR+(SIZE+29) DIV 30+DISKADR;
    END;
END PROGDESCBLDR;

```

```

05254000 T 0385
05254500 T 0387
05255000 T 0387

```

```

COMMENT DOTSNTAX ANALYSES THE SYNTAX OF A PARTIAL WORD DESIGNATOR,
        IT REPORTS IF AN ERROR IS FOUND. IT RETURNS WITH THE
        LITERALS INVOLVED;
BOOLEAN PROCEDURE DOTSNTAX(FIRST,SECOND);
PRT(551) = DOTSNTAX
        INTEGER FIRST,SECOND;
        BEGIN
            LABEL EXIT;

            IF STEPI = LFTBRKET THEN
            IF STEPI = LITNO THEN
            IF STEPI = COLON THEN
            IF STEPI = LITNO THEN
            IF STEPI = RTBRKET THEN
COMMENT IF TESTS ARE PASSED THEN SYNTAX IS CORRECT;
            IF (FIRST + ELBAT[I-3].ADDRESS) *
                (SECOND + ELBAT[I-1].ADDRESS) # 0 THEN
            IF FIRST + SECOND ≤ 48 THEN
COMMENT IF TESTS ARE PASSED THEN RANGES OF LITERALS ARE O.K.;
                GO TO EXIT;
            ERR(114); COMMENT ERROR IF SYNTAX OR RANGE FAILS;
            DOTSNTAX ← TRUE; EXIT; END DOTSNTAX;

```

```

05267000 T 0387
05268000 T 0387
05269000 T 0387
05270000 T 0387

05271000 T 0387
05272000 T 0387
05273000 T 0387
START OF SEGMENT ***** 38
05274000 T 0000
05275000 T 0001
05276000 T 0002
05277000 T 0004
05278000 T 0005
05279000 T 0007
05280000 T 0007
05281000 T 0009
05282000 T 0012
05283000 T 0014
05284000 T 0014
05285000 T 0015
05286000 T 0015
38 IS 20 LONG, NEXT SEG 3

```

```

05287000 T 0387
05288000 T 0387
05289000 T 0387
05290000 T 0387
05291000 T 0387
05292000 T 0387
05293000 T 0387
05294000 T 0387
05295000 T 0387
05296000 T 0387
05297000 T 0387

```

```

05298000 T 0387
05299000 T 0387
05300000 T 0387
05301000 T 0387
05302000 T 0387
05303000 T 0387
05304000 T 0387

```

```

05305000 T 0392

```

```

        BOOLEAN PROCEDURE RANGE(LOWER,UPPER);
PRT(552) = RANGE
        VALUE LOWER,UPPER;
        REAL LOWER,UPPER;
        COMMENT RANGE TESTS THE CLASS OF THE ITEM IN ELBAT[I] TO SEE IF
        IT IS GREATER THAN OR EQUAL TO LOWER OR LESS THAN OR EQUAL TO
        UPPER AND SETS RANGE TO TRUE OR FALSE ACCORDINGLY. THE ITEMS
        CLASS MUST BE IN ELCLASS;
        RANGE←ELCLASS ≥ LOWER AND ELCLASS ≤ UPPER;

```

```

COMMENT GET OBTAINS A SYLLABLE FROM EDOC, THE ARRAY INTO WHICH CODE IS

```

```

      EMITTED;
INTEGER PROCEDURE GET(L); VALUE L; REAL L;
BEGIN
  INTEGER STREAM PROCEDURE GETSYL(W,S); VALUE S;

PRT(553) = GETSYL
      BEGIN DI + LOC GETSYL; DI + DI+6;
        SI + W; SI + SI+S; SI + SI+S; DS + 2 CHR END;

      GET+GETSYL(CODE( L DIV 4+1),L,[46:2]); END GET;

COMMENT CALL SWITCH PERFORMS THE FINAL MESS OF GETTING A PROPER DE-
      SCRIPTOR TO THE TOP OF THE STACK;
PROCEDURE CALLSWITCH(H); VALUE H; REAL H;

PRT(554) = CALLSWITCH
      BEGIN EMITV(GNAT(H)); EMIT0(PRTE); EMIT0(LUD) END CALLSWITCH;

PROCEDURE WRITEPRT(PORS,N,GS); VALUE PORS,N,GS; INTEGER PORS,N,GS;

PRT(555) = WRITEPRT
      BEGIN
        LABEL EXIT;

      STREAM PROCEDURE FILLIT(LIN,PORS,CELL,N,IO);

PRT(556) = FILLIT
      VALUE PORS,CELL,N;
      BEGIN
        LOCAL COUNT;
        LABEL M0,M1,M2,M3,M4,M5,M6,M7,XIT;
        SI:=LOC PORS; SI:=SI+3; DI:=LIN; % "PRT" OR "STACK".
        IF SC="P" THEN
          BEGIN DS:=3 CHR; DS:=LIT("("; END
        ELSE BEGIN
          DS:=5 CHR; DS:=LIT("("; SI:=LOC CELL; SI:=SI+5;
          IF SC="6" THEN DS:=2 LIT"F=" ELSE DS:=2 LIT"F+";
          COUNT:=DI; DI:=LOC CELL; DI:=DI+4;
          DS:=11 RESET; DI:=COUNT;
          END;
        SI:=LOC CELL; SI:=SI+4; TALLY:=4; % LOCATION.

```

```

05306000 T 0392
05307000 T 0392
05308000 T 0392
05309000 T 0392
START OF SEGMENT ***** 39
05310000 T 0000
05311000 T 0000

05312000 T 0003
39 IS 12 LONG, NEXT SEG 3

05313000 T 0392
05314000 T 0392
05315000 T 0392
05316000 T 0392

05317000 T 0396
05318000 T 0396
05319000 T 0396
05320000 T 0396
05321000 T 0396
05322000 T 0396
05323000 T 0396
05324000 T 0396
05325000 T 0396
05325010 T 0396

05325020 T 0396
05325030 T 0396
START OF SEGMENT ***** 40
05325040 T 0000

05325050 T 0000
05325060 T 0000
05325070 T 0000
05325080 T 0000
05325090 T 0000
05325100 T 0000
05325110 T 0001
05325120 T 0002
05325130 T 0002
05325140 T 0004
05325150 T 0005
05325160 T 0006
05325170 T 0007
05325180 T 0007

```

```

3(IF SC="0" THEN % DONT PRINT LEADING ZEROES.
  BEGIN SI:=SI+1; TALLY:=TALLY+63 END ELSE JUMP OUT);
COUNT:=TALLY; DS:=COUNT CHR; TALLY:=0; COUNT:=TALLY;
DS:=4 LIT") = "; CELL:=DI; % SAVE OUR PLACE.
CI:=CI+N;
GO M0;
GO M1;
GO M2;
GO M3;
GO M4;
GO M5;
GO M6;
GO M7;
M0: SI:=ID; SI:=SI+2; DI:=LUC COUNT;
DI:=DI+7; DS:=CHR; DI:=CELL; DS:=COUNT CHR;
GO XIT;
M1: DI:=CELL; DS:=19 LIT"*TEMPORARY STORAGE*"; GO XIT;
M2: DI:=CELL;
DS:=36 LIT"*LIST, LABEL, OR SEGMENT DESCRIPTOR*"; GO XIT;
M3: DI:=CELL; DS:=27 LIT"*CASE STATEMENT DESCRIPTOR*"; GO XIT;
M4: DI:=CELL; DS:=19 LIT"*FORMAT DESCRIPTOR*"; GO XIT;
PRT(557) = *LIST, LABEL, OR SEGMENT DESCRIPTOR*
M5: DI:=CELL; DS:=24 LIT"*OUTER BLOCK DESCRIPTOR*"; GO XIT;
PRT(560) = *LIST, LABEL, OR SEGMENT DESCRIPTOR*
M6: DI:=CELL; DS:=20 LIT"*SEGMENT DESCRIPTOR*"; GO XIT;
PRT(561) = *LIST, LABEL, OR SEGMENT DESCRIPTOR*
M7: DI:=CELL; DS:=18 LIT"*LABEL DESCRIPTOR*";
PRT(562) = *LIST, LABEL, OR SEGMENT DESCRIPTOR*
XIT;
END FILLIT;
PRT(563) = *LIST, LABEL, OR SEGMENT DESCRIPTOR*

```

```

BLANKET(14,LIN);
IF N=1 THEN FILLIT(LIN,PORS,GS,0,ACCUM[1]);
ELSE IF N>1 THEN FILLIT(LIN,PORS,GS,0,INFO[N,LINKR,N,LINKC]);
ELSE FILLIT(LIN,PORS,GS,ABS(N),N);
IF NOHEADING THEN DATIME; WRITELINE;
END WRITEPRT;

```

```

COMMENT GETSPACE MAKES ASSIGNMENTS TO VARIABLES AND DESCRIPTORS IN
THE STACK AND PRT. PERMANENT TELLS WHETHER IT IS A
PERMANENTLY ASSIGNED CELL (ALWAYS IN PRT) OR NOT. NON
PERMANENT CELLS ARE EITHER IN STACK OR PRT ACCORDING TO
MODE. CARE IS TAKEN TO REUSE NON PERMANENT PRT CELLS;
INTEGER PROCEDURE GETSPACE(PERMANENT,L); VALUE PERMANENT,L;
BOOLEAN PERMANENT; INTEGER L;
BEGIN LABEL L1,L2,EXIT;
STREAM PROCEDURE DOIT(C,A,I,S); VALUE C,A;

```

PRT(564) = DOIT

```

05325190 T 0007
05325200 T 0008
05325210 T 0011
05325220 T 0012
05325230 T 0013
05325240 T 0013
05325250 T 0014
05325260 T 0014
05325270 T 0014
05325280 T 0014
05325290 T 0015
05325300 T 0015
05325310 T 0015
05325320 T 0015
05325330 T 0016
05325340 T 0018
05325350 T 0018
05325360 T 0022
05325370 T 0023
05325380 T 0028
05325390 T 0033

```

```

05325400 T 0037
05325410 T 0041
05325420 T 0045
05325430 T 0048
05325440 T 0048

```

```

05325450 T 0049
05325460 T 0051
05325470 T 0055
05325480 T 0063
05325490 T 0067
05325500 T 0079

```

40 IS 80 LONG, NEXT SEG 3

```

05326000 T 0396
05327000 T 0396
05328000 T 0396
05329000 T 0396
05330000 T 0396
05331000 T 0396
05333000 T 0396
05334000 T 0396

```

START OF SEGMENT ***** 41
05334100 T 0000

BEGIN LOCAL N;	05334200 T	0000
DI+S; DS+8 LIT" "; SI+S; DS+9 WDS;	05334300 T	0000
SI+I; SI+SI+2; DI+LOC N; DI+DI+7; DS+CHR;	05334400 T	0002
DI+S; SI+LOC C; 2(DS+4 DEC);	05334500 T	0003
SI+I; SI+SI+3; DS+N CHR;	05334600 T	0004
END;	05334700 T	0005
BOOLEAN M,Q;	05343000 T	0005
STACK(F+3) = M		
STACK(F+4) = Q		
INTEGER ROW,COL,GS;	05344000 T	0005
STACK(F+5) = ROW		
STACK(F+6) = COL		
STACK(F+7) = GS		
IF NOT(STREAMTOG AND (LEVEL>2))THEN	05344400 T	0005
IF STEPI=RELOP THEN	05344500 T	0007
BEGIN	05344510 T	0009
IF STEPI>IDMAX	05344520 T	0009
THEN	05344530 T	0010
BEGIN	05344540 T	0010
IF ELCLASS=ADOP	05344550 T	0011
THEN	05344560 T	0011
IF ELBAT[I].ADDRESS=SUBOP	05344570 T	0011
THEN GS=FZERO ELSE GS+512	05344580 T	0013
ELSE	05344590 T	0015
BEGIN GS+0; I+I-1 END;	05344600 T	0016
IF STEPI#LITNO THEN FLAG(51);	05344610 T	0018
IF ELBAT[I].ADDRESS≥512 THEN GS+1024;	05344615 T	0021
GS+GS+ELBAT[I].ADDRESS	05344620 T	0023
END	05344630 T	0024
ELSE	05344640 T	0025
BEGIN	05344650 T	0025
GS+ELBAT[I].ADDRESS;	05344660 T	0028
IF GS=0 THEN FLAG(51);	05344661 T	0029
IF GS≠FZERO AND GS≤1023 THEN GS+=GS;	05344662 T	0031
IF STEPI#ADOP THEN I+I-1 ELSE	05344670 T	0034
BEGIN	05344680 T	0037
STEPIT;	05344690 T	0038
GS+ELBAT[I].ADDRESS+	05344700 T	0038
(IF ELBAT[I-1].ADDRESS=SUBOP	05344710 T	0039
THEN -GS ELSE +GS);	05344720 T	0041
END;	05344730 T	0044
GS+ABS(GS);	05344740 T	0044
END; Q+GS<512 OR GS>1023;	05344750 T	0045
GO TO EXIT	05344760 T	0047
END ELSE I+I-1;	05344770 T	0047
IF MODE = 0 OR PERMANENT	05345000 T	0049
THEN BEGIN	05346000 T	0050
IF PRTIMAX > 1023 THEN FLAG(148);	05347000 T	0051
IF ASTOG THEN FLAG(505);	05348000 T	0053
PRTI +	05349000 T	0054
PRTIMAX+(GS+PRTIMAX)+1;	05350000 T	0054
IF STUFFTOG THEN IF (M+(LEVEL=1 AND KLASSF>19)) OR	05350100 T	0057
(LEVEL≥3 AND ELBAT[I].CLASS=LABELID) THEN BEGIN	05350120 T	0060

```

        IF NOT M THEN
        DOIT(LABELID,GS,INFO((ELBAT[I]),LINKR,
        (ELBAT[I],LINKC+1)),TWXA[0]) ELSE
        DOIT(KLASSF,GS,INFO((LASTINFO+1),LINKR,(LASTINFO+1),LINKC),
        TWXA[0])) WRITE(STUFF,10,TWXA[*]) END; END
    ELSE BEGIN
        IF STACKCTR > 767 THEN FLAG(149);
        STACKCTR ← (GS + STACKCTR)+1; Q ← FALSE;
        GO TO EXIT END;
L2:   IF GS ≥ 512 THEN GS ← GS+1024;
        Q ← TRUE;
EXIT:  GETSPACE ← GS;
        IF GS2NESTCTR AND GS<FZERO THEN NESTCTR←GS+1;
        IF GS > 1023 THEN GS ← GS-1024;
        IF PRTOG THEN WRITEPRT(IF Q THEN "PRT " ELSE "STACK",L,B2D(GS));
        END GETSPACE;

```

```

05350140 T 0063
05350160 T 0064
05350180 T 0067
05350200 T 0069
05350300 T 0074
05369000 T 0079
05370000 T 0080
05371000 T 0082
05372000 T 0084
05373000 T 0085
05374000 T 0087
05375000 T 0088
05375100 T 0089
05376000 T 0093
05376100 T 0095
05378000 T 0100
41 IS 107 LONG, NEXT SEG 3

```

```

REAL PROCEDURE DEPTH(I); VALUE I; REAL I;
PRT(565) = DEPTH
    BEGIN REAL J,K,T,S,M;

```

```

05400000 T 0396
05401000 T 0396
START OF SEGMENT ***** 42

```

```

STACK(F+3) = J
STACK(F+4) = K
STACK(F+5) = T
STACK(F+6) = S
STACK(F+7) = M

```

```

        IF T+NESTPRT[I]<0 THEN
        BEGIN DEPTH←CALL(T,[22:13]-1],[35:13]);
        IF NESTPRT[I],[2:1]=0 THEN NESTCUR←NESTCUR+1;
        NESTPRT[I],[2:1]+1;
        END
    ELSE IF T,[9:13]≠0 THEN DEPTH←T,[9:13]
    ELSE BEGIN M←0; NESTPRT[I]←-T;
        J←T,[22:13]; K←CALL(J-1],[22:13];
        FOR J←J STEP 1 UNTIL K DO
        IF S+DEPTH(CALL(J))>M THEN M←S;
        M←DEPTH+M+CALL(T,[22:13]-1],[35:13];
        IF NESTCUR≠0 THEN
        IF NESTPRT[I],[2:1]=0 THEN ELSE
        BEGIN T←T&M[9:35:13]; NESTCUR←NESTCUR-1 END
        ELSE T←T&M[9:35:13];
        NESTPRT[I]←T;
        END;
    END;

```

```

05402000 T 0000
05402100 T 0002
05402200 T 0007
05402300 T 0011
05402400 T 0014
05403000 T 0014
05404000 T 0016
05405000 T 0020
05406000 T 0026
05407000 T 0027
05409000 T 0034
05409100 T 0040
05409200 T 0041
05409300 T 0044
05409400 T 0047
05409500 T 0050
05410000 T 0051
05411000 T 0051
42 IS 56 LONG, NEXT SEG 3

```

```

PROCEDURE NESTSORT(L,U); VALUE L,U; REAL L,U; FORWARD;
PRT(566) = NESTSORT
    PROCEDURE SORTNEST;
PRT(567) = SORTNEST

```

```

05411100 T 0396
05412000 T 0396

```

```

        BEGIN ARRAY A[0:14];

STACK(F+2) = A

        REAL I,J,K,T;

STACK(F+3) = I
STACK(F+4) = J
STACK(F+5) = K
STACK(F+6) = T

        REAL P,Q;

STACK(F+7) = P
STACK(F+10) = Q

        STREAM PROCEDURE NESTFORM(I,N,L,A); VALUE I,N;
PRT(570) = NESTFORM
        BEGIN LOCAL S;
          DI=A; 15(DS+8 LIT " ");
          DI=LOC S; DI=DI+7; SI=L; SI=SI+10; DS=CHR;
          DI=A; DI=DI+I; A=DI;
          DI=DI+6; DS= S CHR;
          DI=A; SI=LOC N; DS=4 DEC;
          DI=A; DS=3 FILL;
        END;

```

```

        FOR I=PRTBASE STEP 1 UNTIL PRTOP DO
        IF NESTPRT[I]≠0 THEN
          BEGIN SORTPRT[Q]=I; Q=Q+1 END;
        NESTSORT(0,Q+Q-1);
        FOR P=0 STEP 1 UNTIL Q DO
        BEGIN I=SORTPRT[P]; T=NESTPRT[I];
          NESTFORM(0,DEPTH(I),INFO[T,LINKR,T,LINKC],A);
          WRITE(LINE[DBL],15,A[*]);
          J=T,[22:13]; K=CALL[J-1],[22:13];
          FOR J=J STEP 1 UNTIL K DO
          BEGIN I=CALL[J];
            T=NESTPRT[I];
            NESTFORM(32,DEPTH(I),INFO[T,LINKR,T,LINKC],A);
            WRITE(LINE,15,A[*]);
          END;
          WRITE(LINE[DBL]);
        END;
        END;

```

```

        PROCEDURE NESTSORT(L,U); VALUE L,U; REAL L,U;
        BEGIN REAL I,J,K,M;

```

```

STACK(F+2) = I
STACK(F+3) = J
STACK(F+4) = K
STACK(F+5) = M

```

```

        LABEL AGAIN,TOP,BOTTOM,EXIT;
        IF L≠U THEN

```

```

05413000 T 0396
START OF SEGMENT ***** 43
05414000 T 0001

```

```

05414100 T 0001

```

```

05415000 T 0001

```

```

05416000 T 0001
05417000 T 0003
05418000 T 0005
05419000 T 0006
05420000 T 0007
05421000 T 0008
05422000 T 0008
05423000 T 0009

```

```

05424000 T 0009
05425000 T 0011
05425100 T 0012
05425200 T 0017
05425300 T 0019
05425400 T 0021
05426000 T 0023
05427000 T 0028
05428000 T 0032
05429000 T 0037
05430000 T 0039
05430500 T 0042
05431000 T 0043
05432000 T 0048
05433000 T 0052
05434000 T 0054
05435000 T 0058
05436000 T 0061

```

```

43 IS 66 LONG, NEXT SEG 3

```

```

05437000 T 0396
05438000 T 0396
START OF SEGMENT ***** 44

```

```

05439000 T 0000
05440000 T 0000

```

```

      BEGIN M ← (U+L) DIV 2;
      NESTSORT(L,M);
      NESTSORT(M+1,U);
      I ← K+L; J ← M+1;
AGAIN:  IF I > M THEN GO TO TOP;
      IF J > U THEN GO TO BOTTOM;
      GT1 ← NESTPRT[ SORTPRT[I] ] . [ 33:15 ] . LINK;
      GT2 ← NESTPRT[ SORTPRT[J] ] . [ 33:15 ] . LINK;
      IF INFO[GT1, LINKR, (GT1+1), LINKC] . [ 18:30 ] ≤
        INFO[GT2, LINKR, (GT2+1), LINKC] . [ 18:30 ] THEN
        GO TO BOTTOM;
TOP:    SORTPRT[K] . [ 18:15 ] ← SORTPRT[J];
      J ← J+1;
      IF K+K+1 ≤ U THEN GO TO AGAIN ELSE GO TO EXIT;
BOTTOM: SORTPRT[K] . [ 18:15 ] ← SORTPRT[I];
      I ← I+1;
      IF K+K+1 ≤ U THEN GO TO AGAIN ELSE GO TO EXIT;
EXIT:   FOR I ← L STEP 1 UNTIL U DO
      SORTPRT[I] ← SORTPRT[I] . [ 18:15 ];
      END;
END;

```

```

05441000 T 0000
05442000 T 0003
05443000 T 0004
05444000 T 0005
05445000 T 0008
05446000 T 0009
05447000 T 0010
05448000 T 0013
05449000 T 0016
05450000 T 0019
05451000 T 0022
05452000 T 0023
05453000 T 0026
05454000 T 0027
05455000 T 0030
05456000 T 0033
05457000 T 0034
05458000 T 0037
05459000 T 0039
05460000 T 0043
05461000 T 0043

```

44 IS 46 LONG, NEXT SEG 3

```

COMMENT  ROUTINES IN THIS SECTION COMPILE CODE FOR ALL EXPRESSIONS;
COMMENT  AEXP IS THE ARITHMETIC EXPRESSION ROUTINE;
PROCEDURE AEXP;
  BEGIN
    IF ELCLASS = IFV
    THEN BEGIN IF IFEXP ≠ ATYPE THEN ERR(102) END
    ELSE BEGIN ARITHSEC; SIMPARITH END
  END AEXP;

```

```

06000000 T 0396
06001000 T 0396
06002000 T 0396
06003000 T 0396
06004000 T 0396
06005000 T 0396
06006000 T 0399
06007000 T 0401

```

```

COMMENT ARITHSEC COMPILES FIRST PRIMARY IN AN ARITHMETIC EXPRESSION.
      IN PARTICULAR IT HANDLES P, +P, -P, AND -P*Q WHERE P
      AND Q ARE PRIMARIES;
PROCEDURE ARITHSEC;
  BEGIN
    IF ELCLASS = ADOF
    THEN BEGIN
      STEPIT;
      IF ELBAT[I-1].ADDRESS ≠ SUB THEN PRIMARY
      ELSE BEGIN
        PRIMARY;
        ENDTOG ← LINKTOG; EMIT0(CH$);
        LINKTOG ← ENDTOG; ENDTOG ← FALSE END END
      ELSE PRIMARY END ARITHSEC;
    END;

```

```

06008000 T 0401
06009000 T 0401
06010000 T 0401
06011000 T 0401
06012000 T 0401
06013000 T 0401
06014000 T 0402
06015000 T 0403
06016000 T 0403
06017000 T 0406
06018000 T 0407
06021000 T 0407
06022000 T 0409
06023000 T 0410

```

COMMENT SIMPARITH COMPILES SIMPLE ARITHMETIC EXPRESSIONS ON THE

06024000 T 0412


```

        ASSUMPTION THAT AN ARITHMETIC PRIMARY HAS ALREADY BEEN
        COMPILED. IT ALSO HANDLES THE CASE OF A CONCATENATE
        WHERE ACTUALPARAPART CAUSED THE VARIABLE ROUTINE TO
        COMPILE ONLY PART OF A PRIMARY. MOST OF THE WORK OF
        SIMPARITH IS DONE BY ARITHCOMP, AN ARTIFIAL ROUTINE
        WHICH DOES THE HIERARCHY ANALYSIS USING RECURSION.
        ARITHCOMP IS A SUBROUTINE ONLY TO GET THIS RECURSION;
PROCEDURE SIMPARITH;
    BEGIN
        WHILE ELCLASS = AMPERSAND
            DO BEGIN STEPIT; PRIMARY; PARSE END;
        WHILE ELCLASS ≥ EQVOP DO ARITHCOMP END;

COMMENT ARITHCOMP IS THE GUTS OF THE ARITHMETIC EXPRESSION ROUTINE
ANALYSIS. IT CALLS PRIMARY AT APPROPRIATE TIMES AND
EMITS THE ARITHMETIC OPERATORS. THE HIERARCHY ANALYSIS
IS OBTAINED BY RECURSION;
PROCEDURE ARITHCOMP;
    BEGIN INTEGER OPERATOR, OPCLASS;

    DO BEGIN
        OPERATOR ← 1 & ELBAT[I] [36:17:10];
COMMENT THIS SETS UP THE OPERATOR WHICH WILL BE EMITTED. THE HIGH
ORDER TEN BITS OF THE OPERATOR ARE LOCATED IN [17:10]
OF THE ELBAT WORD;
        OPCLASS ← ELCLASS;
        STEPIT; PRIMARY;
        BEGIN
            WHILE OPCLASS < ELCLASS DO ARITHCOMP;
COMMENT THE CLASSES ARE ARRANGED IN ORDER OF HIERARCHY;
            EMIT(OPERATOR);
            EMIT(0); L ← L-1;
            STACKCT ← 1;
        END;
        END UNTIL OPCLASS ≠ ELCLASS END ARITHCOMP;

STACK(F+2) = OPERATOR
STACK(F+3) = OPCLASS

```

```

06025000 T 0412
06026000 T 0412
06027000 T 0412
06028000 T 0412
06029000 T 0412
06030000 T 0412
06031000 T 0412
06032000 T 0412
06033000 T 0412
06034000 T 0412
06035000 T 0412
06036000 T 0415

```

```

06037000 T 0418
06038000 T 0418
06039000 T 0418
06040000 T 0418
06041000 T 0418
06042000 T 0418
START OF SEGMENT ***** 45

```

```

06043000 T 0000
06044000 T 0000
06045000 T 0002
06046000 T 0002
06047000 T 0002
06048000 T 0002
06049000 T 0002
06051000 T 0003
06052000 T 0003
06053000 T 0006
06054000 T 0006
06054100 T 0007
06054150 T 0009
06054200 T 0009
06055000 T 0009
45 IS 14 LONG, NEXT SEG 3

```

```

INTEGER PROCEDURE EXPRSS; BEGIN AEXP; EXPRSS ← ATYPE END;

```

```

06057000 T 0418

```

```

PROCEDURE POLISHER(EXPECT); VALUE EXPECT; REAL EXPECT;
    BEGIN LABEL EXIT;

```

```

        LABEL EL;
        REAL COUNT, T1, T2;

```

```

STACK(F+2) = COUNT
STACK(F+3) = T1

```

```

06060000 T 0422
06061000 T 0422
START OF SEGMENT ***** 46
06061900 T 0000
06062000 T 0000

```

STACK(F+4) = T2
 STACK(F+5) = S
 STACK(F+6) = SSS
 STACK(F+7) = Z
 PRT(571) = WRITEOUT

BOOLEAN S;
 REAL SSS; INTEGER Z;
 STREAM PROCEDURE WHITEOUT(C,N,L); VALUE C,N;
 BEGIN DI ← L; DS ← 2 LIT "S=";
 SI ← LOC C; SI ← SI+7; DS ← CHR;
 SI ← LOC N; DS ← DEC;
 SB(DS+2LIT " ");
 END;

SSS ← STACKCTR;
 IF STEP1 ≠ LEFTPAREN THEN GO TO EXIT;
 DO BEGIN
 IF STEP1 ≥ OPERATORS THEN
 BEGIN T1 ← (T2 + ELBAT[I]).ADDRESS;
 S ← S OR COUNT = T2.[11:3] < 0;
 COUNT ← T2.[14:2]+COUNT-2;
 IF ELCLASS ≥ OPERATOR THEN
 BEGIN IF T1 ≠ 0 THEN EMIT(T1)
 ELSE BEGIN
 T1 ← T2.LINK+2;
 T2 ← T2.INCR+T1;
 FOR T1 ← T1 STEP 1 UNTIL T2 DO
 EMIT(TAKE(T1));
 END;
 END ELSE BEGIN T2 ← ELCLASS;
 IF STEP1 ≠ LITNO THEN
 BEGIN ERR(500); GO TO EXIT END;
 IF T2 = BITOP THEN EMIT(T1&C
 [36:42:6]) ELSE
 IF T2 = HEXOP THEN EMIT(T1&
 (T2+C DIV 6)[36:45:3]&(C-T2×6)
 [39:45:3]) ELSE
 IF T2 = ISOLATE THEN
 BEGIN T2 ← C;
 IF STEP1 ≠ LITNO
 THEN BEGIN ERR(500);
 GO TO EXIT END;
 EMIT(Z+((T2+C-1)DIV 6-C DIV
 6+1)×512+(48-T2-C)MOD 6×64+
 37));
 END END;
 STEPIT;
 S ← S OR COUNT < 0;
 END ELSE BEGIN
 IF ELCLASS = LABELID THEN
 BEGIN T1:=2;

06063000	T	0000
06063500	T	0000
06064000	T	0000
06065000	T	0000
06066000	T	0000
06067000	T	0001
06067500	T	0002
06068000	T	0003
06068500	T	0003
06069000	T	0004
06070000	T	0006
06071000	T	0007
06072000	T	0008
06074000	T	0010
06075000	T	0013
06076000	T	0015
06077000	T	0016
06078000	T	0018
06079000	T	0019
06080000	T	0021
06081000	T	0022
06082000	T	0024
06083000	T	0027
06084000	T	0027
06085000	T	0028
06086000	T	0029
06087000	T	0031
06088000	T	0033
06089000	T	0034
06090000	T	0036
06091000	T	0039
06092000	T	0041
06093000	T	0042
06094000	T	0043
06095000	T	0044
06096000	T	0045
06097000	T	0046
06098000	T	0046
06099000	T	0046
06099100	T	0046
06099200	T	0048
06100000	T	0052
06101000	T	0054
06102000	T	0054
06103000	T	0054
06104000	T	0056
06104100	T	0056
06104200	T	0057

```

EL:          GT4 + TAKE(T2+GIT(ELBAT[I]));
             PUT(L,T2);
             IF GT4 = 0 THEN GT4 + L;
             IF (GT4:=L-GT4)DIV 4 ≥ 128 THEN
               BEGIN GT4:=0;FLAG(50);END;
             EMIT(GT4×4+T1);
             STEPIT;
             END ELSE
             IF ELCLASS ≠ PERIOD THEN AEXP ELSE BEGIN
               T2+0;
               IF STEP1=PERIOD THEN
                 BEGIN T2+1; STEPIT END;
               IF ELCLASS>IDMAX THEN
                 BEGIN ERR(500); GO TO EXIT END;
               IF ELCLASS = LABELID THEN
                 BEGIN T1 + 0; GO TO EL END;
               IF T1 + ELBAT[I].ADDRESS = 0 THEN
                 BEGIN ERR(100); GO TO EXIT END;
               EMITL(T1);
               IF T1>1023 THEN
                 IF T2=0 THEN FLAG(500)
               ELSE EMITO(PRTE);
               STEPIT;
               END; COUNT + COUNT+1;
             END;
             END UNTIL ELCLASS ≠ COMMA;
             IF ELCLASS ≠ RTPAREN THEN
               BEGIN ERR(104); GO TO EXIT END;
             STEPIT;
             IF FALSE THEN
               BEGIN COUNT + COUNT=EXPECT;
               WRITEOUT(IF COUNT < 0 THEN "-" ELSE
                 IF COUNT = 0 THEN " " ELSE "+",
                 ABS(COUNT),LIN[0]);
               WRITELINE;
             END;
             EXIT; STACKCTR + SSS; END;

```

```

06104300 T 0058
06104400 T 0061
06104500 T 0062
06104510 T 0064
06104520 T 0066
06104600 T 0068
06104700 T 0070
06104800 T 0071
06105000 T 0071
06106000 T 0073
06106100 T 0074
06106200 T 0075
06106300 T 0077
06107000 T 0078
06107100 T 0079
06107200 T 0080
06108000 T 0082
06109000 T 0084
06110000 T 0086
06110100 T 0086
06110200 T 0087
06110300 T 0089
06111000 T 0091
06112000 T 0091
06113000 T 0093
06114000 T 0093
06115000 T 0094
06116000 T 0095
06117000 T 0096
06118000 T 0097
06119000 T 0097
06120000 T 0099
06121000 T 0101
06122000 T 0103
06123000 T 0105
06124000 T 0115
06125000 T 0115

```

46 IS 120 LONG, NEXT SEG 3

```

PROCEDURE PRIMARY;
  BEGIN LABEL

```

```

L1, L2, L3, L4, L5, L6, L7, L8, L9, L10,
L11, L12, L13, L14, L15, L16, L17, L18, L19, L20,
L21, L22, L23, L24, L25, L26, L27, L28, L29, L30,
L31, L32, L33, L34, L35, L36, L37, L38, L39;
SWITCH S +
L1, L2, L3, L4, L5, L6, L7, L8, L9, L10,
L11, L12, L13, L14, L15, L16, L17, L18, L19, L20,
L21, L22, L23, L24, L25, L26, L27, L28, L29, L30,
L31, L32, L33, L34, L35, L36, L37, L38, L39;
LABEL EXIT,RP,LDOT,LAMPER;
GO TO S[ELCLASS];
IF ELCLASS = LFTBRKET THEN

```

```

06126000 T 0422
06127000 T 0422
START OF SEGMENT ***** 47
06128000 T 0000
06129000 T 0000
06130000 T 0000
06131000 T 0000
06132000 T 0000
06133000 T 0002
06134000 T 0002
06135000 T 0002
06136000 T 0002
06137000 T 0022
06138000 T 0022
06139000 T 0025

```

```

        BEGIN STEPIT; VARIABLE(FL);
        IF ELCLASS ≠ RTBRKET THEN
            BEGIN ERR(118); GO TO EXIT END;
        STEPIT;
        GO TO LDOT;
    END;
    IF ELCLASS = NOTOP THEN
        BEGIN STEPIT; PRIMARY;
        EMITLNG; EMIT(0); L=L-1;
        GO TO EXIT;
    END;
    IF ELCLASS = UNKNOWNID THEN ERR(100);
L1:L2:L3:L4:L5:L6:L8:L9:L10:L12:L13:L16:L17:L20:L21:L24:L25:L28:L29:
L32:
        ERR(103); GO TO EXIT;
L7:
        SUBHAND(FALSE); GO TO LDOT;
L11:
        IMPFUN; STACKCT + STACKCT-1; GO TO LDOT;
L14:L15:
        STRMPROCSTMT; GO TO LDOT;
L18:L19:
        PROCSTMT(FALSE); GO TO LDOT;
L22:L23:L26:L27:L30:L31:
        VARIABLE(FP); GO TO LAMPER;
L33:L35:
        EMIT(0&ELBAT[I] [36:17:10]); STEPIT; GO TO LAMPER;
L34:L36:
        EMITNUM(C); STEPIT; GO TO LAMPER;
L38:
        POLISHER(1); GO TO LDOT;
L39:
        STEPIT; PRIMARY; STACKCT + STACKCT-1;
        EMIT(LOD); GO TO LDOT;
L37:
        STEPIT; AEXP;
        STACKCT + STACKCT-1;
        IF ELCLASS ≠ RTPAREN THEN
            BEGIN ERR(104); GO TO EXIT END;
        STEPIT;
        LDOT:DOT;
        LAMPER:
            STACKCT + STACKCT + 1;
            WHILE ELCLASS = AMPERSAND DO
                BEGIN STEPIT; PRIMARY; PARSE END;
EXIT: END PRIMARY;

```

```

PROCEDURE IMPFUN;
    BEGIN REAL T1,T2;

```

```

STACK(F+2) = T1
STACK(F+3) = T2

```

```

T1 ← (T2 + ELBAT[I]).ADDRESS;

```

```

06140000 T 0025
06141000 T 0027
06142000 T 0028
06143000 T 0030
06144000 T 0030
06145000 T 0031
06146000 T 0031
06147000 T 0031
06148000 T 0033
06149000 T 0035
06150000 T 0036
06151000 T 0036
06152000 T 0038
06153000 T 0039
06154000 T 0039
06155000 T 0040
06156000 T 0041
06157000 T 0042
06158000 T 0043
06159000 T 0045
06160000 T 0046
06161000 T 0047
06162000 T 0047
06163000 T 0048
06164000 T 0049
06165000 T 0050
06166000 T 0051
06167000 T 0054
06168000 T 0054
06169000 T 0055
06170000 T 0056
06171000 T 0057
06172000 T 0058
06172500 T 0060
06173000 T 0061
06174000 T 0062
06174500 T 0063
06175000 T 0064
06176000 T 0065
06177000 T 0066
06178000 T 0067
06179000 T 0068
06179500 T 0069
06180000 T 0070
06181000 T 0072
06182000 T 0074

```

47 IS 76 LONG, NEXT SEG 3

```

06183000 T 0422
06184000 T 0422
START OF SEGMENT ***** 48

```

```

06185000 T 0000

```

```

PANA;
IF T1 # 0 THEN EMIT(T1)
ELSE BEGIN
T1 + T2.LINK+T2.INCR+1;
T2 + T2.LINK+2;
FOR T2 + T2 STEP 1 UNTIL T1 DO EMIT(TAKE(T2));
END;
END;

```

```

06186000 T 0002
06187000 T 0002
06188000 T 0004
06189000 T 0005
06190000 T 0007
06191000 T 0009
06192000 T 0014
06193000 T 0014
48 IS 17 LONG, NEXT SEG 3

```

```

PROCEDURE SUBHAND(FROM); VALUE FROM; BOOLEAN FROM;
BEGIN LABEL EXIT;

```

```

STACK(F+2) = T1
REAL T1;
T1 + TAKEFRST;
IF ELCLASS # SUBID AND FROM THEN
BEGIN IF STEP1 # ASSIGNOP THEN
BEGIN FLAG(503); GO TO EXIT END;
STEPIT;
AEXP;
EMIT(XCH);
GO TO EXIT;
END;
EMITL((L+6) DIV 4-(T1.[24:12]-1) DIV 4);
EMITB(BBW,BUMPL,T1.[36:12]);
STEPIT;
ADJUST;
EXIT: END SUBHAND;

```

```

06194000 T 0422
06195000 T 0422
START OF SEGMENT ***** 49
06196000 T 0000
06197000 T 0000
06198000 T 0001
06199000 T 0002
06200000 T 0003
06201000 T 0005
06202000 T 0006
06203000 T 0006
06204000 T 0007
06205000 T 0007
06206000 T 0007
06207000 T 0011
06208000 T 0014
06208500 T 0014
06209000 T 0015
49 IS 19 LONG, NEXT SEG 3

```

```

COMMENT IFEXP COMPILES CONDITIONAL EXPRESSIONS. IT REPORTS THE TYPE
OF EXPRESSIONS AS EXPRSS REPORTS;
INTEGER PROCEDURE IFEXP;
BEGIN INTEGER TYPE,THENBRANCH,ELSEBRANCH;

```

```

06292000 T 0422
06293000 T 0422
06294000 T 0422
06295000 T 0422
START OF SEGMENT ***** 50

```

```

STACK(F+3) = TYPE
STACK(F+4) = THENBRANCH
STACK(F+5) = ELSEBRANCH

```

```

IFCLAUSE;
STACKCT + 0;
THENBRANCH + BUMPL;
COMMENT SAVE L FOR LATER FIXUP;
IFEXP + TYPE + EXPRSS; COMMENT COMPILE 1ST EXPRSS;
STACKCT + 0;
ELSEBRANCH + BUMPL;
EMITB(BFC,THENBRANCH,L);
IF ELCLASS # ELSEV THEN ERR(155) ELSE BEGIN
STEPIT;
AEXP; STACKCT + 1;
COMMENT THIS COMPILES PROPER TYPE SECOND EXPRSS;

```

```

06296000 T 0000
06296500 T 0000
06297000 T 0001
06298000 T 0003
06299000 T 0003
06299500 T 0004
06300000 T 0005
06301000 T 0007
06302000 T 0008
06303000 T 0010
06305000 T 0011
06306000 T 0012

```

```

        EMITB(BFW,ELSEBRANCH,L);
        EMIT(1); L = L-1;
COMMENT THIS IS USED BY EMITLNG TO CLEANUP CODE.  COMPARE WITH
        BOUSEC, BOOCOMP, AND RELATION;
END END IFEXP;

```

```

06307000 T 0012
06308000 T 0013
06309000 T 0015
06310000 T 0015
06311000 T 0015
50 IS 19 LONG, NEXT SEG 3

```

```

COMMENT PARSE COMPILES CODE FOR THE CONCATENATE;
PROCEDURE PARSE;
    BEGIN INTEGER FIRST,SECOND,THIRD;

```

```

06312000 T 0422
06313000 T 0422
06314000 T 0422
START OF SEGMENT ***** 51

```

```

STACK(F+2) = FIRST
STACK(F+3) = SECOND
STACK(F+4) = THIRD

```

```

        LABEL EXIT;
        IF ELCLASS = LFTBRKET THEN
        IF STEPI = LITNO THEN
        IF STEPI = COLON THEN
        IF STEPI = LITNO THEN
        IF STEPI = COLON THEN
        IF STEPI = LITNO THEN
        IF STEPI = RTBRKET THEN
COMMENT IF TEST ARE PASSED THEN SYNTAX IS CORRECT;
        IF (FIRST + ELBAT[I-5],ADDRESS) *
            (SECOND + ELBAT[I-3],ADDRESS) *
            (THIRD + ELBAT[I-1],ADDRESS) = 0 THEN
        IF FIRST + THIRD ≤ 48 THEN
        IF SECOND + THIRD ≤ 48 THEN
COMMENT IF TEST ARE PASSED THEN RANGES OF LITERALS ARE O.K.;
        BEGIN
            STEPIT;
            EMITD(SECOND,FIRST,THIRD);
        STACKCT = 1;
        GO TO EXIT END;
        ERR(113); COMMENT ERROR IF SYNTAX OR RANGE FAILS;
EXIT;    END PARSE;

```

```

06315000 T 0000
06316000 T 0000
06317000 T 0000
06318000 T 0002
06319000 T 0003
06320000 T 0005
06321000 T 0006
06322000 T 0008
06323000 T 0009
06324000 T 0009
06325000 T 0012
06326000 T 0014
06327000 T 0017
06328000 T 0019
06329000 T 0020
06330000 T 0020
06331000 T 0021
06332000 T 0021
06332500 T 0023
06333000 T 0023
06334000 T 0024
06335000 T 0025
51 IS 28 LONG, NEXT SEG 3

```

```

COMMENT DOT COMPILES CODE FOR PARTIAL WORD DESIGNATORS, EXCEPT FOR
        THOSE CASES HANDLED BY THE VARIABLE ROUTINE;
PROCEDURE DOT;
    BEGIN INTEGER FIRST,SECOND; LABEL EXIT;

```

```

06336000 T 0422
06337000 T 0422
06338000 T 0422
06339000 T 0422
START OF SEGMENT ***** 52

```

```

STACK(F+2) = FIRST
STACK(F+3) = SECOND

```

```

        IF ELCLASS = PERIOD THEN BEGIN
            IF DOTSYNTAX(FIRST,SECOND) THEN GO TO EXIT;

        EMIT(0,FIRST,SECOND);
        STEPIT;

```

```

06340000 T 0000
06341000 T 0001
06342000 T 0003
06343000 T 0003
06344000 T 0003
06345000 T 0004

```

EXIT: END END DOT;

06346000 T 0004
52 IS 8 LONG, NEXT SEG 3

PROCEDURE IFCLAUSE;
 BEGIN STEPIT; BEXP;
 IF ELCLASS ≠ THENV THEN ERR(116) ELSE STEPIT END IFCLAUSE;
COMMENT PANA COMPILES THE CONSTRUCT: (<ARIT, EXP.>);

06409000 T 0422
06410000 T 0422
06411000 T 0424
06412000 T 0427

PROCEDURE PANAS;
 BEGIN
 IF STEPI ≠ LEFTPAREN THEN ERR(105)
 ELSE BEGIN STEPIT; AEXP; IF ELCLASS ≠ RTPAREN THEN
 ERR(104) ELSE STEPIT END END PANAS;

06413000 T 0427
06414000 T 0427
06415000 T 0427
06416000 T 0429
06417000 T 0432

COMMENT BANA COMPILES THE CONSTRUCT: [<ARITH, EXP.>];
PROCEDURE BANA;
 BEGIN
 IF STEPI ≠ LFTBRKET THEN ERR(117)
 ELSE BEGIN STEPIT; AEXP; IF ELCLASS ≠ RTBRKET THEN
 ERR(118) ELSE STEPIT END END BANA ;

06418000 T 0435
06419000 T 0435
06420000 T 0435
06421000 T 0435
06422000 T 0436
06423000 T 0439

COMMENT THIS SECTION CONTAINS THE STATEMENT ROUTINES;
COMMENT COMPOUNDTAIL COMPILES COMPOUNDTAILS. IT ALSO ELIMINATES
 COMMENTS FOLLOWING ENDS. AFTER ANY ERROR, ERROR MESSAGES
 ARE SUPPRESSED. COMPOUNDTAIL IS PARTIALLY RESPONSIBLE
 FOR RESTORING THE ABILITY TO WRITE ERROR MESSAGES. SOME
 CARE IS ALSO TAKEN TO PREVENT READING BEYOND THE "END.";

07000000 T 0442
07001000 T 0442
07002000 T 0442
07003000 T 0442
07004000 T 0442
07005000 T 0442
07006000 T 0442

PROCEDURE COMPOUNDTAIL;
PRT(572) = COMPOUNDTAIL

 BEGIN LABEL ANOTHER;

 I ← I-1; BEGINCTR ← BEGINCTR+1;
ANOTHER: ERRORTOG ← TRUE; COMMENT ALLOW ERROR MESSAGES;
 STEPIT;
 IF STREAMTOG THEN STREAMSTMT ELSE STMT;
 IF ELCLASS = SEMICOLON THEN GO TO ANOTHER;
 IF ELCLASS ≠ ENDV
 THEN BEGIN
 ERR(119); GO TO ANOTHER END;
 ENDTOG ← TRUE;
 DO STOPDEFINE ← TRUE UNTIL
 STEPI ENDV AND ELCLASS ≥ UNTILV
 OR NOT ENDTOG;
 ENDTOG ← FALSE;
 IF BEGINCTR ← BEGINCTR-1 ≠ 0 EQV ELCLASS = PERIOD

07007000 T 0442
START OF SEGMENT ***** 53
07008000 T 0000
07009000 T 0002
07010000 T 0003
07011000 T 0004
07012000 T 0006
07013000 T 0007
07014000 T 0008
07015000 T 0009
07016000 T 0010
07017000 T 0011
07018000 T 0011
07019000 T 0013
07020000 T 0015
07021000 T 0015

```

        THEN BEGIN
            IF BEGINCTR = 0 THEN
                BEGIN FLAG(143); BEGINCTR + 1; GO ANOTHER END;
FLAG (120);
FCR:= (LCR:=MKABS(CBUFF[9]))-9;
    IF LISTER THEN PRINTCARD;
FCR:= (LCR:=MKABS(TBUFF[9]))-9 END;
    IF ELCLASS = PERIOD THEN
        BEGIN
            GT5 + "ND;END."&"E"[1:43:5];
            MOVE(1,GT5,CBUFF[0]);
            LASTUSED+4;
            ELBAT[I+I-2] +SPECIAL[20];
            ELCLASS + SEMICOLON END
        END COMPOUNDTAIL;

```

```

07022000 T 0017
07023000 T 0019
07024000 T 0019
07025000 P 0022
07025010 C 0023
07025020 C 0026
07025030 C 0042
07026000 T 0045
07027000 T 0045
07028000 T 0046
07029000 T 0048
07030000 T 0049
07031000 T 0050
07032000 T 0052
07033000 T 0053
53 IS 55 LONG, NEXT SEG 3

```

```

        REAL AXNUM;
PRT(573) = AXNUM
        PROCEDURE ACTUALPARAPART(SBIT,INDEX); VALUE SBIT,INDEX;
PRT(574) = ACTUALPARAPART
        BOOLEAN SBIT; REAL INDEX;
        BEGIN LABEL EXIT,COMMON,ANOTHER,POL;
        REAL PCTR,SCLASS,ACCLASS;
STACK(F+2) = PCTR
STACK(F+3) = SCLASS
STACK(F+4) = ACCLASS
        STREAM PROCEDURE WRITEAX(LINE,ACCUM,N,SEQ); VALUE N;
PRT(575) = WRITEAX
        BEGIN DI + LINE; 15(DS + 8 LIT " ");
        DI + LINE; SI + SEQ; SI + SI-16; DS + WDS;
        DI + DI+4; DS + 20 LIT "ACCIDENTAL ENTRY AT ";
        SI + ACCUM; SI + SI+3; DS + N CHR;
        SI + SEQ; DI + SEQ; DI + DI-16; DS + WDS;
        END;

```

```

07034000 T 0442
07035000 T 0442
07036000 T 0442
07037000 T 0442
START OF SEGMENT ***** 54
07038000 T 0000
07038100 T 0000
07038200 T 0000
07038300 T 0002
07038400 T 0003
07038500 T 0006
07038600 T 0007
07038700 T 0008

```

```

        BOOLEAN VBIT,IDBIT;
STACK(F+5) = VBIT
STACK(F+6) = IDBIT
        PCTR + 1;
        ANOTHER: ACCLASS + STEPI&0[47:47:1];
        STACKCT + 0;
        GT1 + TAKE(INDEX+PCTR);
        VBIT + BOOLEAN(GT1,V0);
        SCLASS + GT1,CLASS&0[47:47:1];
        IF VBIT THEN BEGIN AEXP; GO TO COMMON END;
        IF SBIT THEN SCLASS + NAMEID;
        IDBIT + BOUID < ACCLASS AND ACCLASS < LABELID;
        IF SCLASS = NAMEID THEN
            BEGIN

```

```

07039000 T 0008
07040000 T 0008
07041000 T 0009
07041200 T 0012
07042000 T 0012
07043000 T 0014
07044000 T 0015
07045000 T 0018
07046000 T 0019
07047000 T 0021
07048000 T 0023
07049000 T 0024

```



```

POL:      IF IDBIT THEN VARIABLE(FL)
          ELSE
            IF ELCLASS = POLISHV THEN POLISHER(1)
            ELSE ERR(IF ELCLASS=0 THEN 0 ELSE 123);
            GO TO COMMON;
          END;
        IF SCLASS = REALARRAYID THEN
          IF ACLASS = REALARRAYID THEN
            BEGIN VARIABLE(FL); GO TO COMMON END
          ELSE GO TO POL;
        IF SCLASS ≠ REALID THEN
          BEGIN FLAG(503);
            AEXP;
            ERRORTOG ← TRUE;
            GO TO COMMON;
          END;
        GT1 ← TABLE(I+1);
        IF GT1 = COMMA OR GT1 = RTPAREN THEN
          BEGIN IF IDBIT THEN
            BEGIN IF ACLASS = REALID AND
              BOOLEAN(ELBAT[I].FORMAL) THEN BEGIN
                CHECKER (ELBAT[I]);
                EMITPAIR(ELBAT[I],ADDRESS,LOD);
                STEPIT; END
              ELSE VARIABLE(FL);
              GO TO COMMON END;
            IF ELCLASS ≤ STRNGCON AND ELCLASS > LABELID
              THEN BEGIN PRIMARY; GO TO COMMON END;
          END;
        EMIT0(NOP); EMIT0(NOP);
        SCLASS ← L;
        ADJUST;
        ACLASS ← L.[36:10];
        IF IDBIT THEN
          BEGIN VARIABLE(FL);
            IF ELCLASS < AMPERSAND THEN GO TO COMMON;

            SIMPARITH;
            END ELSE AEXP;
        IF LISTER THEN
          BEGIN ACCUM[1] ← Q;
            WRITEAX(LIN[0],ACCUM[1],Q.[12:6],
              INFO[LASTSEQR0W, LASTSEQUENCE]);
            WRITELINE;
          END;
        AXNUM ← AXNUM+1;
        EMIT0(RTS);
        EMITB(BFW,SCLASS,L);
        EMITNUM(ACCLASS);
        EMITPAIR(TAKE(PROINFO),ADDRESS,LOD);
        EMIT0(INX);
        EMITN(512);
        EMITD(33,18,15);
        EMIT(0);
        EMITD(5,5,1);
COMMON:  PCTR ← PCTR+1;
        IF ELCLASS = COMMA THEN GO TO ANOTHER;

```

```

07050000 T 0024
07051000 T 0025
07052000 T 0026
07053000 T 0028
07054000 T 0032
07055000 T 0032
07056000 T 0032
07057000 T 0033
07058000 T 0034
07059000 T 0036
07060000 T 0036
07061000 T 0037
07062000 T 0038
07063000 T 0039
07064000 T 0039
07065000 T 0040
07066000 T 0040
07067000 T 0042
07068000 T 0043
07069000 T 0044
07070000 T 0045
07070500 T 0047
07071000 T 0048
07072000 T 0050
07073000 T 0050
07074000 T 0052
07075000 T 0052
07076000 T 0053
07077000 T 0055
07078000 T 0055
07079000 T 0057
07080000 T 0058
07081000 T 0058
07082000 T 0059
07083000 T 0060
07084000 T 0061
07084500 T 0062
07085000 T 0062
07086000 T 0063
07086100 T 0064
07086200 T 0064
07086300 T 0066
07086400 T 0068
07086500 T 0069
07086600 T 0079
07086700 T 0079
07087000 T 0081
07088000 T 0081
07089000 T 0083
07090000 T 0083
07091000 T 0085
07092000 T 0086
07093000 T 0087
07093100 T 0088
07093200 T 0089
07094000 T 0090
07095000 T 0092

```

```

        IF ELCLASS ≠ RTPAREN THEN
            BEGIN ERR(129); GO TO EXIT END;
        IF TAKE(INDEX).NODIMPART+1 ≠ PCTR THEN
            BEGIN ERR(128); GO TO EXIT END;
        STEPIT;
        STACKCT ← 0;
EXIT:    END ACTUAL PARAPART;

```

```

07096000 T 0093
07097000 T 0094
07098000 T 0096
07099000 T 0098
07100000 T 0100
07100500 T 0100
07101000 T 0101
54 IS 106 LONG, NEXT SEG 3

```

```

PROCEDURE PROCSTMT(FROM); VALUE FROM; BOOLEAN FROM;
BEGIN
    REAL HOLE, ADDRESS;

```

```

STACK(F+2) = HOLE
STACK(F+3) = ADDRESS

```

```

STACK(F+4) = J

```

```

REAL J; LABEL OK;

```

```

LABEL EXIT;
SCATTERELBAT;
HOLE ← ELBAT[I];
ADDRESS ← ADDR5F;
IF NESTOG THEN
    IF MODE ≠ 0 THEN
        IF TABLE(I+1) ≠ ASSIGNOP THEN
            BEGIN FOR J ← CALLINFO STEP 1 UNTIL CALLX DO
                IF CALL[J] = ADDRESS THEN GO TO OK;
            CALL[CALLX+CALLX+1] ← ADDRESS;

```

```

OK;

```

```

END;
CHECKER(HOLE);
IF ELCLASS ≠ PROCID THEN
    IF NOT FORMALF THEN
        IF TABLE(I+1) = ASSIGNOP THEN
            BEGIN VARIABLE(2 ← REAL(FROM)); GO TO EXIT END;

```

```

PRT(576) = *LIST, LABEL, OR SEGMENT DESCRIPTOR*

```

```

COMMENT CALL VARIABLE TO HANDLE THIS ASSIGNMENT OPERATION;

```

```

IF ELCLASS ≠ PROCID EQV FROM
    THEN BEGIN ERR(159); GO TO EXIT END;

```

```

COMMENT IT IS PROCEDURE IF AND ONLY WE COME FROM STMT;

```

```

STEPIT;
EMIT0(MKS);
IF ELCLASS = LEFTPAREN
    THEN ACTUALPARAPART(FALSE, GIT(HOLE))
    ELSE IF FORMALF THEN L ← L-1
    ELSE IF TAKE(GIT(HOLE)).NODIMPART ≠ 0 THEN ERR(128);
EMITV(ADDRESS);

```

```

EXIT:    END PROCSTMT;

```

```

07391000 T 0442
07392000 T 0442
07393000 T 0442
START OF SEGMENT ***** 55

```

```

07393100 T 0000
07394000 T 0000
07395000 T 0000
07396000 T 0000
07397000 T 0001
07397100 T 0002
07397200 T 0002
07397210 T 0003
07397300 T 0006
07397400 T 0008
07397500 T 0014
07397600 T 0018
07398000 T 0019
07399000 T 0019
07400000 T 0020
07401000 T 0021
07402000 T 0023

```

```

07403000 T 0028
07404000 T 0028
07405000 T 0029
07406000 T 0031
07407000 T 0031
07408000 T 0032
07409000 T 0032
07410000 T 0033
07411000 T 0035
07412000 T 0037
07413000 T 0042
07425000 T 0042

```

```

55 IS 46 LONG, NEXT SEG 3

```

```

PROCEDURE STRMPROCSTMT;
BEGIN REAL WHOLE, FIX, T1;

```

```

07426000 T 0442
07427000 T 0442
START OF SEGMENT ***** 56

```

STACK(F+2) = WHOLE
 STACK(F+3) = FIX
 STACK(F+4) = T1

```

    WHOLE = ELBAT(I); FIX = -1;
    IF ELCLASS ≠ STRPROCID THEN EMIT(0);
    IF WHOLE, LVL ≠ 1 THEN
      BEGIN FIX = L; L = L+1 END;
    EMIT(MKS);
    T1 = TAKEFRST,[1:6];
    FOR GT1 = 1 STEP 1 UNTIL T1 DO EMIT(0);
    IF STEPI ≠ LEFTPAREN THEN ERR(128)
    ELSE BEGIN ACTUALPARAPART(TRUE,GIT(WHOLE));
      IF FIX < 0 THEN EMITV(WHOLE,ADDRESS)
      ELSE BEGIN T1 = L; L = FIX;
        WHOLE = TAKE(GIT(WHOLE));
        EMITNUM(T1+2=WHOLE,[16:12]);
        L = T1;
        EMITB(BBW,BUMPL,WHOLE,[28:12]);
      END;
    END END STRMPROCSTMT;
  
```

```

07428000 T 0000
07429000 T 0000
07430000 T 0000
07431000 T 0002
07432000 T 0004
07433000 T 0005
07434000 T 0007
07435000 T 0008
07436000 T 0010
07437000 T 0014
07438000 T 0015
07439000 T 0018
07440000 T 0019
07441000 T 0022
07442000 T 0024
07443000 T 0026
07444000 T 0027
07445000 T 0030
07446000 T 0030
56 IS 33 LONG, NEXT SEG 3
  
```

```

INTEGER PROCEDURE BAE;
  BEGIN BAE = BUMPL; CONSTANTCLEAN; ADJUST END BAE;
  
```

```

07458000 T 0442
07459000 T 0442
  
```

```

COMMENT RELSESTMT COMPILES THE RELEASE STATEMENT;
COMMENT DOSTMT HANDLES THE DO STATEMENT;
PROCEDURE DOSTMT;
  
```

PRT(577) = DOSTMT

```

  BEGIN INTEGER TL;
  
```

STACK(F+2) = TL

```

  FOULED = L;
  
```

```

    STEPIT; TL=L; STMT; IF ELCLASS ≠ UNTILV THEN ERR(131)
    ELSE BEGIN
      STEPIT; BEXP; EMITB(BBC,BUMPL,TL) END
    END DOSTMT;
  
```

```

07460000 T 0447
07481000 T 0447
07482000 T 0447
07483000 T 0447
START OF SEGMENT ***** 57
07483500 T 0000
07484000 T 0000
07485000 T 0000
07486000 T 0004
07487000 T 0005
07488000 T 0008
57 IS 11 LONG, NEXT SEG 3
  
```

```

COMMENT WHILESTMT COMPILES THE WHILE STATEMENT;
PROCEDURE WHILESTMT;
  
```

PRT(600) = WHILESTMT

```

  BEGIN INTEGER BACK,FRONT;
  
```

STACK(F+2) = BACK

```

07489000 T 0447
07490000 T 0447
07491000 T 0447
START OF SEGMENT ***** 58
  
```

STACK(F+3) = FRONT

FOULED + L;

STEPIT; BACK + L; BEXP; FRONT + BUMPL;
IF ELCLASS # DOV THEN ERR(132) ELSE
BEGIN STEPIT; STMT; EMITB(BBW,BUMPL,BACK);
CONSTANTCLEAN; EMITB(BFC,FRONT,L) END END WHILESTMT;

07491500 T 0000
07492000 T 0000
07493000 T 0000
07494000 T 0004
07495000 T 0006
07496000 T 0010
58 IS 14 LONG, NEXT SEG 3

COMMENT GOSTMT COMPILES GO TO STATEMENTS. GOSTMT LOOKS AT THE
EXPRESSION. IF IT IS SIMPLE ENOUGH WE GO DIRECTLY.
OTHERWISE A CALL ON THE MCP IS GENERATED IN ORDER TO GET
STORAGE RETURNED. SEE DEXP AND GENGO;

PROCEDURE GOSTMT;

PRT(601) = GOSTMT

BEGIN

REAL ELBW;

STACK(F+2) = ELBW

LABEL GOMCP,EXIT;
IF STEPI = TOV THEN STEPIT;
IF ELCLASS = LABELID THEN TB1 = TRUE
ELSE IF ELCLASS = SWITCHID THEN TB1 = FALSE
ELSE BEGIN IF ELCLASS = POLISHV THEN
BEGIN POLISHER(1); EMITB(BFW) END
ELSE ERR(501);
GO TO EXIT
END;
IF NOT LOCAL(ELBAT[1]) THEN
BEGIN
IF TB1 THEN
BEGIN EMITV(GNAT(ELBAT[1]));
EMITB(BFW);
STEPIT;
GO TO EXIT END;
BEGIN ERR(501); GO TO EXIT END;
END;
IF TB1 THEN BEGIN GOGEN(ELBAT[1],BFW); STEPIT;
CONSTANTCLEAN; GO EXIT END
ELSE BEGIN
ELBW = ELBAT[1];

BANA;
EMITB(DUP);
EMITB(ADD);
EMITB(BFW);
GT3 = TAKE(GT4+GIT(ELBW))+GT4;
FOR GT4 = GT4+1 STEP 1 UNTIL GT3 DO
GOGEN(TAKE(GT4),BFW);
END;
EXIT; END GOSTMT;

07497000 T 0447
07498000 T 0447
07499000 T 0447
07500000 T 0447
07501000 T 0447

07502000 T 0447
07503000 T 0447
START OF SEGMENT ***** 59

07504000 T 0000
07505000 T 0000
07506000 T 0002
07507000 T 0003
07511000 T 0006
07512000 T 0007
07513000 T 0009
07514000 T 0011
07515000 T 0011
07516000 T 0011
07516100 T 0012
07516200 T 0013
07516300 T 0013
07516400 T 0015
07516500 T 0016
07516600 T 0016
07517000 T 0017
07517500 T 0018
07518000 T 0018
07519000 T 0021
07520000 T 0022
07521000 T 0022
07522000 T 0023
07523000 T 0023
07524000 T 0024
07525000 T 0024
07526000 T 0025
07527000 T 0026
07528000 T 0029
07529000 T 0033
07530000 T 0035
07531000 T 0035
59 IS 39 LONG, NEXT SEG 3

```

PROCEDURE GOGEN(LABELBAT,BRANCHTYPE);
  VALUE LABELBAT,BRANCHTYPE;
  REAL LABELBAT,BRANCHTYPE;
  BEGIN
    IF BOOLEAN(GT1+TAKE(GT2+GIT(LABELBAT))).[1:1]
      THEN EMITB(BRANCHTYPE,BUMPL,GT1,[36:12])
    COMMENT LABELR SETS THE SIGN OF THE ADDITIONAL INFO FOR A LABEL
      NEGATIVE WHEN THE LABEL IS ENCOUNTERED. SO THIS MEANS
      THAT WE NOW KNOW WHERE TO GO;
      ELSE BEGIN EMIT(GT1); EMIT(BRANCHTYPE);
        PUT(GT1&L[36:36:12],GT2) END END GOGEN;

```

```

07535000 T 0447
07536000 T 0447
07537000 T 0447
07538000 T 0447
07539000 T 0447
07540000 T 0450
07541000 T 0453
07542000 T 0453
07543000 T 0453
07544000 T 0453
07545000 T 0456

```

```

COMMENT SIMPGO IS USED ONLY BY THE IF STMT ROUTINE, IT DETERMINES IF
  A STATEMENT IS A SIMPLE GO TO STATEMENT;
BOOLEAN PROCEDURE SIMPGO;
  BEGIN LABEL EXIT;

```

```

    IF ELCLASS = GOV
      THEN BEGIN
        IF STEPI = TOV THEN STEPIT;
        IF ELCLASS = LABELID THEN
          IF LOCAL(ELBAT[I]) THEN
            BEGIN SIMPGO = TRUE; GO EXIT END;
            I = I+1; ELCLASS = GOV END;
      END;
  EXIT; END SIMPGO;

```

```

07546000 T 0458
07547000 T 0458
07548000 T 0458
07549000 T 0458
START OF SEGMENT ***** 60
07550000 T 0000
07551000 T 0000
07552000 T 0001
07553000 T 0003
07554000 T 0004
07555000 T 0005
07556000 T 0007
07557000 T 0009
60 IS 13 LONG, NEXT SEG 3

```

```

COMMENT IFSTMT COMPILES IF STATEMENTS. SPECIAL CARE IS TAKEN TO
  OPTIMIZE CODE IN THE NEIGHBORHOOD OF THE JUMPS. TO SOME
  EXTENT SUPERFLUOUS BRANCHING IS AVOIDED;
PROCEDURE IFSTMT;
  BEGIN REAL T1,T2; LABEL EXIT;

```

```

07558000 T 0458
07559000 T 0458
07560000 T 0458
07561000 T 0458
07562000 T 0458
START OF SEGMENT ***** 61

```

```

STACK(F+2) = T1
STACK(F+3) = T2

```

```

  IFCLAUSE;
  IF SIMPGO
    THEN BEGIN
      T1 = ELBAT[I];
    IF STEPI = ELSEV
      THEN BEGIN
        STEPIT;
        IF SIMPGO
          THEN BEGIN
            GOGEN(ELBAT[I],BFC); GOGEN(T1,BFW);
          STEPIT; GO TO EXIT END ELSE BEGIN EMITLNG;GOGEN(T1,BFC);
            STMT ; GO TO EXIT END END ;
            EMITLNG; GOGEN(T1,BFC);
            GO EXIT END;
          T1 = BUMPL; STMT;
        IF ELCLASS = ELSEV THEN

```

```

07563000 T 0000
07564000 T 0000
07565000 T 0000
07566000 T 0001
07567000 T 0002
07568000 T 0003
07569000 T 0004
07570000 T 0004
07571000 T 0004
07572000 T 0005
07573000 T 0007
07574000 T 0010
07575000 T 0011
07576000 T 0013
07577000 T 0013
07578000 T 0016

```

```

        BEGIN IF L-T1>1023 THEN ADJUST; EMITB(BFC,T1,L);
                GO EXIT END;
STEPIT;
IF SIMPGO
    THEN BEGIN
        T2 ← L; L ← T1-2; GOGEN(ELBAT[I],BFC); L ← T2;
        STEPIT; GO EXIT END;
T2 ← BUMPL; CONSTANTCLEAN;
IF L-T1>1023 THEN ADJUST; EMITB(BFC,T1,L); STMT;
IF L-T2>1023 THEN ADJUST; EMITB(BFW,T2,L);
EXIT; END IFSTMT;

```

```

07579000 T 0016
07579100 T 0020
07580000 T 0021
07581000 T 0021
07582000 T 0021
07583000 T 0022
07584000 T 0026
07585000 T 0027
07585100 T 0030
07586000 T 0034
07587000 T 0037
61 IS 41 LONG, NEXT SEG 3

```

```

COMMENT LABELR HANDLES LABELED STATEMENTS. IT PUTS L INTO THE
        ADDITIONAL INFO AND MAKES ITS SIGN NEGATIVE. IT COMPILES
        AT THE SAME TIME ALL THE PREVIOUS FORWARD REFERENCES SET
        UP FOR IT BY GOGEN. (THE ADDITIONAL INFO LINKS TO A LIST
        IN THE CODE ARRAY OF ALL FORWARD REFERENCES);

```

PROCEDURE LABELR;

PRT(602) = LABELR

BEGIN LABEL EXIT, ROUND;

DEFINE ELBATWORD=RR9#,LINK=GT2#,INDEX=GT3#,ADDITIONAL
=GT4#,NEXTLINK=GT5#;

REAL OLDL;

STACK(F+2) = OLDL

```

DO BEGIN OLDL ← L;
    IF STEPI ≠ COLON THEN
        BEGIN ERR(133); GO TO EXIT END;
    IF NOT LOCAL(ELBATWORD + ELBAT[I-1])
        THEN BEGIN FLAG(134); GO TO ROUND END;
    IF STEPI = COLON THEN
        BEGIN I ← I-1; ADJUST END ELSE
    IF ELCLASS = LITND THEN L ← 4×C ELSE
    IF ELCLASS=ASTRISK THEN
        BEGIN IF MODE ≠ 0 OR ASTOG THEN
            FLAG(505);
            ASTOG ← TRUE;
            L ← 4×PRTI;
        END ELSE
        I ← I-2;
    IF STEPI ≠ COLON THEN
        BEGIN ERR(133); GO TO EXIT END;
    IF L < OLDL THEN
        BEGIN FLAG(504); GO TO ROUND END;
GT1 ← TABLE(I+1);
LINK ← (ADDITIONAL + TAKE(INDEX + GIT(ELBATWORD)))
        ,[36:12]);
    IF ADDITIONAL < 0 THEN
        BEGIN FLAG(135); GO TO ROUND END;
FOULED ← L;
    IF TABLE(I+1) = COLON THEN
        BEGIN
            IF LINK≠0 THEN BEGIN OLDL ← L;

```

```

07588000 T 0458
07589000 T 0458
07590000 T 0458
07591000 T 0458
07592000 T 0458
07593000 T 0458
07594000 T 0458
START OF SEGMENT ***** 62
07595000 T 0000
07596000 T 0000
07596500 T 0000
07597000 T 0000
07597500 T 0000
07598000 T 0001
07599000 T 0003
07600000 T 0004
07600100 T 0007
07600200 T 0008
07600300 T 0010
07600400 T 0013
07600410 T 0015
07600420 T 0016
07600430 T 0018
07600440 T 0018
07600450 T 0020
07600500 T 0020
07600600 T 0021
07600700 T 0022
07600800 T 0024
07600900 T 0025
07600950 T 0027
07601000 T 0028
07602000 T 0030
07603000 T 0032
07604000 T 0032
07604010 T 0034
07604020 T 0035
07604030 T 0037
07604040 T 0037

```

```

DO BEGIN NEXTLINK ← GET(LINK);
  L ← LINK;
  IF OLDL.[36:10] = L.[36:10] ≥ 128
  THEN FLAG(50) ELSE
    EMIT(OLDL = LINK & 0[46:46:2] +
      0 & NEXTLINK[46:46:2] + 3072);
  L ← L-1;
  END UNTIL LINK = LINK = NEXTLINK DIV 4 = L;
L ← OLDL; END; STEPIT;
DO IF STEP1 ≤ STRNGCON AND ELCLASS ≥
  NONLITNO THEN EMITWORD(C)
  ELSE BEGIN ERR(500); I ← I-1 END
UNTIL STEP1 ≠ COMMA;
I ← I-1;
END ELSE
  WHILE LINK ≠ 0
  DO BEGIN
    NEXTLINK ← GET(LINK-2);
    IF L = LINK > 1023 THEN ADJUST;
    EMITB(GET(LINK-1), LINK, L);
    LINK ← NEXTLINK END;
  PUT(=ADDITIONAL & L[36:36:12], INDEX);
ROUND: ERROR TOG ← TRUE END UNTIL STEP1 ≠ LABELID;
EXIT; END LABELR;

```

```

07604050 T 0039
07604060 T 0041
07604067 T 0042
07604068 T 0043
07604070 T 0045
07604080 T 0048
07604085 T 0050
07604090 T 0051
07604100 T 0054
07604110 T 0055
07604120 T 0057
07604130 T 0058
07604140 T 0063
07604150 T 0064
07604160 T 0065
07605000 T 0065
07606000 T 0067
07607000 T 0068
07607100 T 0070
07608000 T 0072
07609000 T 0074
07610000 T 0075
07645000 T 0078
07646000 T 0080
62 IS 84 LONG, NEXT SEG 3

```

```

PROCEDURE FILLSTMT(SIZE); VALUE SIZE; INTEGER SIZE;
PRT(603) = FILLSTMT
  BEGIN
  COMMENT "COCT" PERFORMS THE OCTAL CONVERT FOR THE FILL STATEMENT.
  IF THERE ARE ANY NON-OCTAL DIGITS, THIS PROCEDURE RETURNS
  A ZERO AND THEN THE 3 LOW-ORDER BITS OF THE BAD DIGIT ARE
  RESET AND IGNORED AND ERROR NUMBER 303 IS PRINTED. "COCT"
  ALLOWS FLAG BITS TO BE SET, WHEREAS "OCTIZE" DOES NOT.
  N      NUMBER OF CHARACTERS TO BE CONVERTED.
  SKBIT  NUMBER OF BITS TO SKIP BEFORE STARTING CONVERSION.
  THIS IS BECAUSE THE NO. OF CHARS. MAY BE LESS THAN
  8 AND IT MUST BE RIGHT-JUSTIFIED IN CD(CODEFILE).
  ACC    ADDRESS OF THE ACCUM WHERE ALPHA INFO IS KEPT.
  ,
  REAL STREAM PROCEDURE COCT(N, SKBIT, ACC, CD); VALUE N, SKBIT;
PRT(604) = COCT
  BEGIN
  SI := ACC; SI := SI + 6; DI := CD; DS := 8 LIT "00000000";
  DI := CD; SKIP SKBIT DB; TALLY := 1;
  N(IF SC > "7" THEN TALLY := 0; SKIP 3 SB;
  3(IF SB THEN DS := 1 SET ELSE SKIP 1 DB; SKIP 1 SB));
  COCT := TALLY
  END COCT;

```

```

07647000 T 0458
07647500 T 0458
07648000 T 0458
07648500 T 0458
07649000 T 0458
07649500 T 0458
07650000 T 0458
07650500 T 0458
07651000 T 0458
07651500 T 0458
07652000 T 0458
07652500 T 0458
07653000 T 0458
07653500 T 0458
START OF SEGMENT ***** 63
07654000 T 0000
07654500 T 0000
07655000 T 0002
07655500 T 0003
07656000 T 0005
07656500 T 0007
07657000 T 0007
07657500 T 0008

```

REAL T2;

STACK(F+2) = T2

PRT(605) = ZEERO

LABEL L1;
STREAM PROCEDURE ZEERO(D);

BEGIN
DI:=D;DS:=8 LIT"00000000";
SI:=D;31(32(DSI=WDS)); DS:=30 WDS;
END ZEERO;

STREAMTOG:=BOOLEAN(2);
SEGMENTSTART(TRUE);
IF STEPI≠ASSIGNOP THEN ZEERO(CODE(1))
ELSE BEGIN
FOR T2:=1 STEP 1 UNTIL SIZE DO
BEGIN
IF STEPI>IDMAX THEN
BEGIN
IF ELCLASS≠LITNO AND ELCLASS≠NONLITNO THEN
IF ELCLASS≠STRNGCON THEN
IF ELCLASS=ADOP AND
(STEPI=NONLITNO OR ELCLASS=LITNO) THEN
C:=C & ELBAT[I-1][1:21:1]
ELSE BEGIN ERROR(302); GO TO L1 END;
IF ELCLASS=STRNGCON AND COUNT=8 THEN
MOVECHARACTERS(8,ACCUM[1],3,CODE(T2),0)
ELSE MOVE(1,C,CODE(T2))
END
ELSE IF COUNT≤19 AND ACCUM[1],[18:18]="OCT" THEN
BEGIN
IF COCT(COUNT=3,48-(COUNT=3)×3,ACCUM[1],
CODE(T2))=0 THEN FLAG(303)
END
ELSE BEGIN ERROR(302); GO TO L1 END;
IF STEPI≠COMMA THEN GO TO L1
END;
ERROR(54);
END;
L1:
RIGHT(SIZE×4);
STREAMTOG:=FALSE;
SEGMENT(SIZE,0);
PROGDESCBLDR(ADDRSF,TRUE,SIZE,DDES);
END FILLSTMT;

PROCEDURE STMT;
BEGIN LABEL

L1, L2, L3, L4, L5, L6, L7, L8, L9, L10,
L11, L12, L13, L14, L15, L16, L17, L18, L19, L20,
L21, L22, L23, L24, L25, L26, L27, L28, L29, L30,

07658000 T 0008
07658500 T 0008

07659000 T 0008
07659500 T 0009
07660000 T 0010
07660500 T 0012

07661000 T 0012
07661500 T 0013
07662000 T 0014
07662500 T 0019
07663000 T 0020
07663500 T 0021
07664000 T 0021
07664500 T 0022
07665000 T 0022
07665500 T 0024
07666000 T 0025
07666500 T 0026
07667000 T 0029
07667500 T 0031
07668000 T 0033
07668500 T 0035
07669000 T 0040
07669500 T 0044
07670000 T 0045
07670500 T 0048
07671000 T 0048
07671500 T 0052
07672000 T 0057
07672500 T 0057
07673000 T 0060
07673500 T 0061
07674000 T 0064
07674500 T 0064
07675000 T 0064
07675500 T 0065
07676000 T 0066
07676500 T 0067
07677000 T 0068
07677500 T 0069

63 IS 72 LONG, NEXT SEG 3

07711000 T 0458
07712000 T 0458
START OF SEGMENT ***** 64
07713000 T 0000
07714000 T 0000
07715000 T 0000


```

L31, L32, L33, L34, L35, L36, L37, L38, L39, L40,
L41, L42, L43, L44, L45, L46, L47, L48, L49, L50,
L51, L52, L53, L54;
SWITCH S +
L1, L2, L3, L4, L5, L6, L7, L8, L9, L10,
L11, L12, L13, L14, L15, L16, L17, L18, L19, L20,
L21, L22, L23, L24, L25, L26, L27, L28, L29, L30,
L31, L32, L33, L34, L35, L36, L37, L38, L39, L40,
L41, L42, L43, L44, L45, L46, L47, L48, L49, L50,
L51, L52, L53, L54;
LABEL AGAIN, EXIT;
STACKCT + 0;
AGAIN: GO TO S[ELCLASS];
IF ELCLASS = COLON THEN
    BEGIN STEPIT; GT1 + L;
    IF ELCLASS = COLON THEN
        BEGIN ADJUST; I + I-1 END
    ELSE IF ELCLASS = LITNO THEN L + 4 * C
    ELSE I + I-1;
    IF L < GT1 OR STEP1 ≠ COLON THEN
        BEGIN ERR(504); GO TO EXIT END;
    STEPIT;
    GO TO AGAIN;
END;
IF ELCLASS = 0 THEN FLAG(100); FLAG(145);
L1:L2:L3:L4:L5:L6:L9:L11:L13:L14:L15:L16:L17:L20:L21:L25:L28:L29:L24:
L33:L34:L35:L36:L37:L39:
ERR(144); GO TO EXIT;
L7:L8:
SUBHAND(TRUE); GO TO EXIT;
L10:L18:L19:
PROCSTMT(TRUE); GO TO EXIT;
L12:
STRMPROCSTMT; GO TO EXIT;
L22:L23:L26:L27:L30:L31:
VARIABLE(FS); GO TO EXIT;
L32:
LABELR; GO TO AGAIN;
L38:
POLISHER(0); GO TO EXIT;
L40:
IF ELBAT[I], ADDRESS = STREAMV THEN
    BEGIN INLINE; GO TO EXIT END;
FLAG(146);
IF TABLE(I-2) = ENDOV AND MODE > 0 THEN
    BEGIN I + I-2; ELCLASS + ENDOV; GO TO EXIT END;
I + I-1; ERRORLOG + TRUE; BLOCK(FALSE);
ELCLASS + TABLE(I+I-1); GO TO EXIT;
L42:
DBLSTMT; GO TO EXIT;
L43:
FORSTMT; GO TO EXIT;
L44:
WHILESTMT; GO TO EXIT;
L45:
DOSTMT; GO TO EXIT;
L51:

```

```

07716000 T 0000
07717000 T 0000
07718000 T 0000
07719000 T 0000
07720000 T 0002
07721000 T 0002
07722000 T 0002
07723000 T 0002
07724000 T 0002
07725000 T 0002
07726000 T 0030
07726990 T 0030
07727000 T 0031
07727010 T 0034
07727020 T 0034
07727030 T 0036
07727040 T 0037
07727050 T 0039
07727060 T 0041
07727070 T 0044
07727080 T 0046
07727090 T 0048
07727100 T 0048
07727110 T 0049
07728000 T 0049
07729000 T 0051
07730000 T 0052
07732000 T 0052
07732000 T 0053
07733000 T 0054
07734000 T 0055
07735000 T 0056
07736000 T 0057
07737000 T 0058
07738000 T 0059
07739000 T 0059
07740000 T 0060
07741000 T 0061
07742000 T 0062
07743000 T 0062
07744000 T 0063
07745000 T 0064
07746000 T 0065
07747000 T 0067
07748000 T 0067
07749000 T 0070
07750000 T 0073
07751000 T 0076
07752000 T 0079
07753000 T 0079
07754000 T 0080
07755000 T 0080
07756000 T 0081
07757000 T 0081
07758000 T 0082
07759000 T 0082
07760000 T 0083

```

```

                IFSTMT; GO TO EXIT;
L52:            GOSTMT; GO TO EXIT;
L53:            IOSTMT; GO TO EXIT;
L54:            IF STEPI = DECLARATORS THEN
BEGIN
                IF ELBAT[I].ADDRESS = STREAMV THEN IF STEPI =
                LEFTPAREN THEN
BEGIN
                    ELCLASS+TABLE(I+I-1) ;
                    COMPOUNDTAIL ;
                    GO TO EXIT ;
                END ELSE I + I = 1;
                I + I = 1;
                BLOCK(FALSE);      END ELSE COMPOUNDTAIL;
L46:L47:L48:L50:
L49:L41:
EXIT: END STMT;

```

07761000 T 0083
 07762000 T 0084
 07763000 T 0084
 07764000 T 0085
 07765000 T 0085
 07766000 T 0086
 07767000 T 0086
 07768000 T 0087
 07768100 T 0087
 07768110 T 0090
 07768120 T 0090
 07768130 T 0091
 07768140 T 0093
 07768160 T 0093
 07768170 T 0094
 07768180 T 0096
 07768200 T 0097
 07769000 T 0099
 07770000 T 0099
 07771000 T 0099
 64 IS 100 LONG, NEXT SEG 3

```

PROCEDURE IOSTMT;
    IF STEPI ≠ LITND OR (GT1+ELBAT[I].ADDRESS)>15 THEN ERR(98)ELSE
    BEGIN EMIT(ELBAT[I-1].ADDRESS&GT1[41:47:1]&GT1[36:44:3]);
        STEPIT
    END SCOPE STATEMENT;

```

07991000 T 0458
 07993000 T 0458
 07994000 T 0458
 07995000 T 0463
 07996000 T 0468
 07997000 T 0468

```

PROCEDURE FORSTMT;
BEGIN
    OWN REAL B,STMTSTART,REGO,RETURNSTORE,ADDRES,V,VRET,

```

08008000 T 0468
 08009000 T 0468
 08010000 T 0468
 START OF SEGMENT ***** 65

PRT(606) = B
 PRT(607) = STMTSTART
 PRT(610) = REGO
 PRT(611) = RETURNSTORE
 PRT(612) = ADDRES
 PRT(613) = V
 PRT(614) = VRET

```

                BRET;
PRT(615) = BRET
    OWN BOOLEAN SIGNA,SIGNB,SIGNC, INT,

```

08011000 T 0000
 08012000 T 0000

PRT(616) = SIGNA
 PRT(617) = SIGNB
 PRT(620) = SIGNC
 PRT(621) = INT

CONSTANA,CONSTANB,CONSTANC;

08013000 T 0000

PRT(622) = CONSTANA
 PRT(623) = CONSTANB
 PRT(624) = CONSTANC

DEFINE SIMPLEB = SIGNC#;	FORMALV = SIGNA#;	08014000	T	0000
SIMPLEV = CONSTANA#;	A = V#;	08015000	T	0000
OPDC = TRUE#;	DESC = FALSE#;	08016000	T	0000
K = BRET#;		08017000	T	0000
LABEL EXIT;		08018000	T	0000
COMMENT PLUG EMITS EITHER AN OPERAND CALL ON A VARIABLE OR A CALL ON A	CONSTANT DEPENDING ON THE REQUIREMENTS;	08019000	T	0000
PROCEDURE PLUG(C,A);	VALUE C,A; REAL A; BOOLEAN C;	08020000	T	0000
PRT(625) = PLUG				
IF C THEN EMITNUM(A) ELSE EMITV(A,ADDRESS);		08021000	T	0000
COMMENT SIMPLE DETERMINES IF AN ARITHMETIC EXPRESSION IS + OR = A		08022000	T	0003
CONSTANT OR A SIMPLE VARIABLE. IT MAKES A THROUGH REPORT		08023000	T	0003
ON ITS ACTIVITY. IT ALSO MAKES PROVISION FOR THE RESCAN		08024000	T	0003
OF ELBAT (THIS IS THE ACTION WITH K = SEE CODE IN THE		08025000	T	0003
TABLE ROUTINE FOR FURTHER DETAILS);		08026000	T	0003
BOOLEAN PROCEDURE SIMPLE(B,A,S);	BOOLEAN B,S; REAL A;	08027000	T	0003
PRT(626) = SIMPLE				
BEGIN		08028000	T	0003
S = IF STEP1 ≠ ADOP THEN FALSE ELSE ELBAT[I],ADDRESS		08029000	T	0003
≠ SUB;		08030000	T	0006
IF ELCLASS = ADOP THEN STEP1;		08031000	T	0008
IF ELCLASS ≥ NONLITNO AND ELCLASS ≤ STRNGCON		08032000	T	0010
THEN BEGIN K = K+1; SIMPLE = TRUE;		08033000	T	0011
ELBAT[I] = O&COMMENTV[2:41:7]&K[16:37:11];		08034000	T	0014
INFO[0,K] = A + C; B = TRUE END		08035000	T	0017
ELSE BEGIN		08036000	T	0021
B = FALSE; A = ELBAT[I];		08037000	T	0022
SIMPLE = REALID ≤ ELCLASS AND ELCLASS ≤ INTID END;		08038000	T	0024
STEP1 END SIMPLE;		08039000	T	0026
COMMENT TEST EMITS THE STEP-UNTIL ELEMENT TEST;		08040000	T	0029
PROCEDURE TEST;		08041000	T	0029
PRT(627) = TEST				
BEGIN		08042000	T	0029
IF NOT CONSTANB THEN		08043000	T	0029
BEGIN EMITO(SUB); IF SIMPLEB THEN EMITV(B,ADDRESS)		08044000	T	0030
ELSE BEGIN		08045000	T	0032
EMITL(2+L-BRET);		08046000	T	0034
EMITB(8BW,BUMPL,B);		08047000	T	0036
END;		08048000	T	0038
EMITO(MUL); EMIT(0) END;		08049000	T	0038
EMITO(IF SIGNB THEN GEQ ELSE LEQ); EMIT(0); L=L-1		08050000	T	0039
END TEST;		08051000	T	0043
BOOLEAN PROCEDURE SIMPI(ALL);	VALUE ALL; REAL ALL;	08052000	T	0044
PRT(630) = SIMPI				
BEGIN		08053000	T	0044
CHECKER(VRET=ALL);		08054000	T	0044

```

ADDRESS ← ALL.ADDRESS;
FORMALV ← ALL.[912] = 2;
IF T ← ALL.CLASS > INTARRAYID OR T < BOUID OR
  GT1 ← (T=BOUID) MOD 4 < 1 THEN
  ERR(REAL(T ≠ 0) × 51 + 100);
INT ← GT1 = 2;
SIMPI ← T ≤ INTID END SIMPI;

```

```

08055000 T 0046
08056000 T 0047
08057000 T 0049
08058000 T 0052
08059000 T 0054
08060000 T 0057
08061000 T 0058

```

```

COMMENT STORE EMITS THE CODE FOR THE STORE INTO THE FOR INDEX;
PROCEDURE STORE(S); VALUE S; BOOLEAN S;

```

PRT(631) = STORE

```

BEGIN
  IF FORMALV THEN BEGIN EMIT0(XCH); S ← FALSE END
  ELSE BEGIN
    EMITL(ADDRESS);
    IF ADDRESS > 1023 THEN EMIT0(PRTE) END;
    T ← (REAL(S)+1)×16;
    EMIT0((IF INT THEN T+512 ELSE 4×T)+4) END STORE;
  END

```

```

08062000 T 0062
08063000 T 0062
08064000 T 0062
08065000 T 0062
08066000 T 0065
08067000 T 0065
08068000 T 0066
08069000 T 0068
08070000 T 0070

```

```

COMMENT CALL EFFECTS A CALL ON THE INDEX;
PROCEDURE CALL(S); VALUE S; BOOLEAN S;

```

PRT(632) = CALL

```

BEGIN
  IF SIMPLV
  THEN IF S THEN EMITV(ADDRESS) ELSE EMITN(ADDRESS)
  ELSE BEGIN
    EMITL(2+L-VRET);
    EMITB(BBW,BUMPL,V);
    IF S THEN EMIT0(LOD) END END CALL;
  END

```

```

08071000 T 0074
08072000 T 0074
08073000 T 0074
08074000 T 0074
08075000 T 0075
08076000 T 0078
08077000 T 0079
08078000 T 0080
08079000 T 0083

```

```

PROCEDURE FORLIST(NUMLE); VALUE NUMLE; BOOLEAN NUMLE;

```

PRT(633) = FORLIST

```

BEGIN
  PROCEDURE FIX(STORE,BACK,FORWART,START);

```

PRT(634) = FIX

```

  VALUE STORE,BACK,FORWART,START;
  REAL STORE,BACK,FORWART,START;
  BEGIN
    EMITB(GET(FORWART-1),FORWART,START);
    IF RETURNSTORE ≠ 0
    THEN BEGIN
      L ← STORE; EMITNUM(B-BACK);
      EMITPAIR(RETURNSTORE,STD) END END FIX;
  END

```

```

08080000 T 0084
08081000 T 0084
08082000 T 0084
START OF SEGMENT ***** 66
08083000 T 0000
08084000 T 0000
08085000 T 0000
08086000 T 0000
08087000 T 0002
08088000 T 0002
08089000 T 0003
08090000 T 0005

```

```

INTEGER BACKFIX, FORWARDBRANCH, FOOT, STOREFIX;

```

```

08091000 T 0006

```

STACK(F+2) = BACKFIX
 STACK(F+3) = FORWARDBRANCH
 STACK(F+4) = FOOT
 STACK(F+5) = STOREFIX

```

    LABEL BRNCH,EXIT;
    STOREFIX ← L; Q ← REAL(MODE=0)+3;
    FOR K ← 1 STEP 1 UNTIL Q DO EMIT0(NOP);
    IF NUMLE
      THEN BEGIN
        BACKFIX ← L;
        IF FORMALV THEN CALL(DESC) END
      ELSE BACKFIX ← V + REAL(SIMPLEV)-1;

    AEXP;
    COMMENT PICK UP FIRST ARITHMETIC EXPRESSION;
    IF ELCLASS = STEPV
      THEN BEGIN
    COMMENT HERE WE HAVE A STEP ELEMENT;
        BACKFIX ← BUMPL;
    COMMENT LEAVE ROOM FOR FORWARD JUMP;
        IF FORMALV THEN CALL(DESC); CALL(OPDC);
    COMMENT FETCH INDEX;
        IF I > 70 THEN BEGIN NXTELBT ← 1; I ← 0 END
        ELSE REGO ← 1;
        IF SIMPLEB ← SIMPLE(CONSTANB,B,SIGNB) AND
          (ELCLASS = UNTILV OR ELCLASS = WHILEV)
          THEN BEGIN
    COMMENT WE HAVE A SIMPLE STEP FUNCTION;
            PLUG(CONSTANB,B);
          END ELSE BEGIN
    COMMENT THE STEP FUNCTION IS NOT SIMPLE; WE CONSTRUCT A
            SUBROUTINE;
            I ← IF I < 4 THEN 0 ELSE REGO; STEPIT;
            SIGNB ← CONSTANB ← FALSE;
            EMIT(0); B ← L;
            AEXP; EMIT0(XCH);
            BRET ← L;
            EMIT0(BFW) END;
            EMIT0(REAL(SIGNB)*32+ADD);
            EMIT0(BFW,BACKFIX,L);
            IF ELCLASS = UNTILV
              THEN BEGIN COMMENT STEP=UNTIL ELEMENT;
                STORE(TRUE); IF FORMALV THEN CALL(OPDC);
                STEPIT; AEXP; TEST END
              ELSE BEGIN COMMENT STEP=WHILE ELEMENT;
                IF ELCLASS ≠ WHILEV THEN
                  BEGIN ERR(153); GO TO EXIT END;
                STEPIT; STORE(FALSE); BEXP END END
            ELSE BEGIN
    COMMENT WE DO NOT HAVE A STEP ELEMENT;
                STORE(FALSE);
                IF ELCLASS = WHILEV
                  THEN BEGIN
    COMMENT WE HAVE A WHILE ELEMENT;
                    STEPIT; BEXP END
                  ELSE BEGIN
    COMMENT ONE EXPRESSION ELEMENT;

```

08092000	T	0006
08093000	T	0006
08094000	T	0009
08095000	T	0014
08096000	T	0014
08097000	T	0014
08098000	T	0015
08099000	T	0017
08100000	T	0019
08101000	T	0019
08102000	T	0019
08103000	T	0019
08104000	T	0020
08105000	T	0021
08106000	T	0021
08107000	T	0022
08108000	T	0022
08109000	T	0025
08110000	T	0025
08111000	T	0027
08112000	T	0029
08113000	T	0030
08114000	T	0031
08115000	T	0033
08116000	T	0033
08117000	T	0034
08118000	T	0034
08119000	T	0034
08120000	T	0034
08121000	T	0038
08122000	T	0039
08123000	T	0040
08124000	T	0042
08125000	T	0042
08126000	T	0043
08127000	T	0045
08128000	T	0046
08129000	T	0046
08130000	T	0047
08131000	T	0050
08132000	T	0051
08133000	T	0052
08134000	T	0052
08135000	T	0054
08136000	T	0056
08137000	T	0056
08138000	T	0056
08139000	T	0057
08140000	T	0057
08141000	T	0058
08142000	T	0058
08143000	T	0059
08144000	T	0060

```

                IF ELCLASS # COMMA THEN BEGIN
                    EMITB(BFW,BUMPL,L+2); BACKFIX + L END
                ELSE BACKFIX + L + 2;
                L + L+1; EMIT(BFW); GO TO BRNCH END END;
COMMENT THIS IS THE COMMON POINT;
IF ELCLASS = COMMA THEN EMITLNG; L + L+1;
EMIT(BFC);
BRNCH: FORWARDBRANCH + L; DIALA + DIALB + 0;
IF ELCLASS = COMMA
    THEN BEGIN
        STEPIT;
        FORLIST(TRUE);
        FIX(STOREFIX,BACKFIX,FORWARDBRANCH,STMTSTART) END
    ELSE BEGIN
        IF ELCLASS # DOV
            THEN BEGIN ERR(154); REGO+L; GO EXIT END;
        STEPIT;
        IF NUMLE THEN FOOT := GETSPACE(FALSE,1); % TEMP.
        STMT;

        IF NUMLE THEN BEGIN
            EMITV(RETURNSTORE + FOOT); EMITC(BBW) END
        ELSE BEGIN
            EMITB(BBW,BUMPL,BACKFIX); RETURNSTORE + 0 END;
            STMTSTART + FORWARDBRANCH; B + L;
            CONSTANTCLEAN; REGO + L;
            FIX(STOREFIX,BACKFIX,FORWARDBRANCH,L) END;
EXIT:  END FORLIST;

```

```

08145000 T 0060
08146000 T 0061
08147000 T 0065
08148000 T 0066
08149000 T 0069
08150000 T 0069
08151000 T 0072
08152000 T 0073
08153000 T 0075
08154000 T 0075
08155000 T 0076
08156000 T 0076
08157000 T 0077
08158000 T 0079
08159000 T 0079
08160000 T 0079
08161000 T 0082
08162000 T 0083
08163000 T 0085
08164000 T 0086
08165000 T 0086
08166000 T 0087
08167000 T 0089
08168000 T 0089
08169000 T 0092
08170000 T 0094
08171000 T 0095
08172000 T 0096
66 IS 100 LONG, NEXT SEG 65

```

```

REAL T1,T2,T3,T4;

STACK(F+2) = T1
STACK(F+3) = T2
STACK(F+4) = T3
STACK(F+5) = T4

```

```

NXTELBT + 1; I + 0;
STEPIT;
IF SIMPI(VRET+ELBAT[I])
    THEN BEGIN
        IF STEPI # ASSIGNOP THEN BEGIN ERR(152); GO EXIT END;
        T1 + L; IF FORMALV THEN EMITN(ADDRES);
        K + 0;
        IF SIMPLE(CONSTANA,A,SIGNA) THEN
            IF ELCLASS = STEPV THEN
                IF SIMPLE(CONSTANB,B,SIGNB) THEN
                    IF ELCLASS = UNTILV THEN
                        IF SIMPLE(CONSTANC,Q,SIGNC) THEN
                            IF ELCLASS = DOV THEN
                                BEGIN
                                    PLUG(CONSTANA,A);
                                    IF SIGNA THEN EMITC(CHS);
                                    RETURNSTORE + BUMPL; ADJUST; CONSTANTCLEAN;
                                    STMTSTART + L;
STEPIT;

```

```

08173000 T 0084

08174000 T 0084
08175000 T 0086
08176000 T 0087
08177000 T 0087
08178000 T 0089
08179000 T 0091
08180000 T 0094
08181000 T 0094
08182000 T 0096
08183000 T 0097
08184000 T 0099
08185000 T 0100
08186000 T 0102
08187000 T 0103
08188000 T 0103
08189000 T 0104
08190000 T 0106
08191000 T 0109
08192000 T 0109

```

```

T1 ← (((4096 × RETURNSTORE+STMTSTART)×2+
REAL(CONSTANB))×2+
REAL(CONSTANC))×2+
REAL(SIGNB))×2+
REAL(SIGNC));
T2 ← VRET;
T3 ← B;
T4 ← Q;
STMT;
SIGNC ← BOOLEAN(T1.[47:1]);
SIGNB ← BOOLEAN(T1.[46:1]);
CONSTANC ← BOOLEAN(T1.[45:1]);
CONSTANB ← BOOLEAN(T1.[44:1]);
STMTSTART ← T1.[32:12];
RETURNSTORE ← T1.[20:12];
VRET ← T2;
B ← T3;
Q ← T4;
SIMPLEV ← SIMPI(VRET);
IF FORMALV THEN EMITN(ADDRES); EMITV(ADDRES);
PLUG(CONSTANB,B);
EMITC(IF SIGNB THEN SUB ELSE ADD);
EMITB(BFW,RETURNSTORE,L);
STORE(TRUE);
IF FORMALV THEN CALL(OPDC);
PLUG(CONSTANC,Q);
IF SIGNC THEN EMITC(CHS);
SIMPLEB ← TRUE; TEST; EMITLNG;
EMITB(BBC,BUMPL,STMTSTART);
GO TO EXIT END;
I ← 2; K ← 0;
SIMPLEV ← SIMPI(VRET);
V ← T1 END
ELSE BEGIN
EMIT(0); V ← L; SIMPLEV ← FALSE; FORMALV ← TRUE;
VARIABLE(FR); EMITC(XCH); VRET ← L; EMITC(BFW);
IF ELCLASS≠ASSIGNOP THEN BEGIN ERR(152); GO EXIT END;
END;
STEPIT; FORLIST(FALSE); L ← REGO;
EXIT; K ← 0 END FORSTMT;

```

```

08193000 T 0110
08194000 T 0112
08195000 T 0113
08196000 T 0114
08197000 T 0115
08198000 T 0116
08199000 T 0116
08200000 T 0117
08201000 T 0118
08202000 T 0118
08203000 T 0120
08204000 T 0121
08205000 T 0122
08206000 T 0123
08207000 T 0125
08208000 T 0126
08209000 T 0127
08210000 T 0127
08211000 T 0128
08212000 T 0129
08213000 T 0132
08214000 T 0133
08215000 T 0135
08216000 T 0136
08217000 T 0137
08218000 T 0138
08219000 T 0139
08220000 T 0141
08221000 T 0143
08222000 T 0145
08223000 T 0147
08224000 T 0148
08225000 T 0149
08226000 T 0150
08227000 T 0151
08228000 T 0154
08229000 T 0157
08230000 T 0159
08231000 T 0159
08232000 T 0161
65 IS 165 LONG, NEXT SEG 3

```

```

REAL PROCEDURE REED;
PRT(635) = REED
BEGIN
LABEL EOF; INTEGER I,J,K;

```

```

STACK(F+3) = I
STACK(F+4) = J
STACK(F+5) = K

```

```

PRT(636) = MOVE

```

```

STREAM PROCEDURE MOVE(N,F,T); VALUE N,T;
BEGIN SI:=F; DI:=T; DS:=N WDS END MOVE;

```

```

08999000 T 0468
08999025 T 0468
08999050 T 0468
START OF SEGMENT ***** 67
08999075 T 0000
08999100 T 0000

```

```

J:=1;
READ(CODISK[NO])[EOF];
PRT(637) = EOF
REED:=I:=FETCH(MKABS(CODISK(1)));
K:=MKABS(CODE(0))=1;
WHILE I=J>30 DO
  BEGIN
    MOVE(30,CODISK(0),K); K:=K+30; J:=J+30;
    READ(CODISK);
  END;
MOVE(I=J,CODISK(0),K);
READ(CODISK)[EOF];
EOF;
END REED;

```

```

08999125 T 0001
08999150 T 0003

08999175 T 0008
08999200 T 0014
08999225 T 0019
08999250 T 0020
08999275 T 0020
08999300 T 0027
08999325 T 0031
08999350 T 0031
08999375 T 0036
08999400 T 0041
08999425 T 0042

```

67 IS 47 LONG, NEXT SEG 3

```

PROCEDURE RIGHT(L); VALUE L; INTEGER L;
  BEGIN
    INTEGER I,J;

    I:=(L+7) DIV 4;
    MOVE(1,I,CODISK(0));
    MOVE(29,CODE(0),CODISK(1));
    WRITE(CODISK);
    J:=29;
    WHILE I=J>0 DO
      BEGIN
        MOVE(30,CODE(J),CODISK(0));
        WRITE(CODISK);
        J:=J+30;
      END;
    END RIGHT;

```

```

08999450 T 0468
08999475 T 0468
08999500 T 0468

```

START OF SEGMENT ***** 68

```

08999525 T 0000
08999550 T 0001
08999575 T 0005
08999600 T 0012
08999625 T 0016
08999650 T 0017
08999675 T 0019
08999700 T 0019
08999725 T 0026
08999750 T 0030
08999775 T 0031
08999800 T 0032

```

68 IS 85 LONG, NEXT SEG 3

```

COMMENT THE PROGRAM ROUTINE DOES THE INITIALIZATION AND THE WRAPUP
FOR THE REST OF THE COMPILER. THE MAIN PROGRAM OF THE COMPILER
IS SIMPLY A CALL ON THE PROGRAM ROUTINE;
PROCEDURE PROGRAM;
PRT(640) = PROGRAM
  BEGIN
    STREAM PROCEDURE MDESC(WD,TOLOC);VALUE WD;

    BEGIN DI+LOC WD; DS+ SET;SI+ LOC WD; DI+TOLOC;DS+WDS END;
PRT(641) = MDESC

```

```

09000000 T 0468
09001000 T 0468
09002000 T 0468
09003000 T 0468

```

```

09004000 T 0468
09005000 T 0468

```

START OF SEGMENT ***** 69

```

09006000 T 0000

```



```

        DEFINE STARTINTRSC=426#;
        LABEL L1;
        LISTOG:=LISTER:=BOOLEAN(1=ERRORCOUNT.[46:1]);
COMMENT  LISTOG IS NOT SET BY DEFAULT ON TIMESHARING;
        NOHEADING := TRUE;
        ERRORCOUNT := 0;
        ERRMAX:=999; % MAY BE CHANGED IN DOLLARCARD,
        BASENUM:=10000; ADDVALUE:=1000; NEWBASE:=TRUE;
COMMENT  DEFAULT VALUES FOR "SSEQ" OPTION;
        LASTUSED := 4; % FOR INITIALIZATION.
        NEXTINFO ← LASTINFO ← LASTSEQROW×256+LASTSEQUENCE+1;
        PUTNBUMP(0);
        GT1 ← " "
        MDESC(GT1,INFO[ LASTSEQROW, LASTSEQUENCE]);
        BLANKET(0,INFO[ LASTSEQROW, LASTSEQUENCE]); % FOR "S CHECK".
        READACARD; % INITIALIZATION OF NCR,FCR, AND LCR, AND
        % READS FIRST CARD INTO CARD BUFFER.
        LASTUSED := 1; % ASSUMES CARD ONLY UNTIL TOLD DIFFERENTLY.
        NXTELBT ← 1;
        PRTI←PRTIMAX←PRTBASE;
        MRCLEAN ← TRUE;
COMMENT  START FILLING TABLES NEEDED TO COMPILE A PROGRAM;
        FILL TEN[*] WITH

```

```

OCT1771110463422054,    OCT1761332600326467,    OCT1751621340414205,
OCT1742165630517247,    OCT1732623176643120,    OCT1723370036413744,
OCT1714266046116735,    OCT1705343457542525,    OCT1676634373473252,
OCT1651040347241213,    OCT1641250441111455,    OCT1631522551333770,
OCT1622047303622767,    OCT1612461164567564,    OCT1603175421725521,
OCT1574034726313046,    OCT1565044113775657,    OCT1556255136775233,
OCT1547730366574502,    OCT1521171646433362,    OCT1511430220142257,
OCT1501736264172732,    OCT1472325741231521,    OCT1463013331500045,
OCT1453616220020057,    OCT1444561664024072,    OCT1435716241031111,
OCT1427301711237333,    OCT1401116227350722,    OCT1371341675243107,
OCT1361632254513731,    OCT1352200727636717,    OCT1342641115606502,
OCT1333411341150223,    OCT1324313631402270,    OCT1315376577702746,
OCT1306676337663537,    OCT1261045602764047,    OCT1251257143561061,
OCT1241532774515275,    OCT1232061573640554,    OCT1222476132610706,
OCT1213215561353071,    OCT1204061115645707,    OCT1175075341217270,
OCT1166314631463146,    OCT1141000000000000,    OCT1131200000000000,
OCT1121440000000000,    OCT1111750000000000,    OCT1102342000000000,
OCT1073032400000000,    OCT1063641100000000,    OCT1054611320000000,
OCT1045753604000000,    OCT1037346545000000,    OCT1011124027620000,
OCT0001351035564000,    OCT0011643245121000,    OCT0022214116345200,
OCT0032657142036440,    OCT0043432772446150,    OCT0054341571157602,
OCT0065432127413543,    OCT0076740555316473,    OCT0111053071060221,
OCT0121265707274266,    OCT0131543271153343,    OCT0142074147406234,
OCT0152513201307703,    OCT0163236041571663,    OCT0174105452130240,
OCT0205126764556310,    OCT0216354561711772,    OCT0231004771627437,
OCT0241206170175347,    OCT0251447626234641,    OCT0261761573704011,
OCT0272356132665013,    OCT0303051561442216,    OCT0313664115752661,
OCT0324641141345435,    OCT0336011371636745,    OCT0347413670206536,
OCT0361131664625027,    OCT0371360241772234,    OCT0401654312370703,
OCT0412227375067064,    OCT0422675274304701,    OCT043454553366062,
OCT0444367706263476,    OCT0455465667740415,    OCT0467003245730521,
OCT0501060411731665,    OCT0511274514320242,    OCT0521553637404312,
OCT0532106607305375,    OCT0542530351166674,    OCT0553256443424453,

```

```

09024000 T 0001
09025000 T 0001
09028000 T 0001
09028010 T 0005
09028050 T 0005
09028900 T 0006
09028910 T 0006
09028920 T 0007
09028930 T 0009
09029000 T 0009
09033000 T 0010
09034000 T 0013
09034100 T 0014
09034200 T 0015
09034500 T 0017
09035000 T 0019
09036000 T 0019
09037000 T 0019
09038000 T 0020
09039000 T 0021
09040000 T 0022
09040100 T 0023
09041000 T 0023
09042000 T 0023
START OF SEGMENT *****
09043000 T 0024
09044000 T 0024
09045000 T 0024
09046000 T 0024
09047000 T 0024
09048000 T 0024
09049000 T 0024
09050000 T 0024
09051000 T 0024
09052000 T 0024
09053000 T 0024
09054000 T 0024
09055000 T 0024
09056000 T 0024
09057000 T 0024
09058000 T 0024
09059000 T 0024
09060000 T 0024
09061000 T 0024
09062000 T 0024
09063000 T 0024
09064000 T 0024
09065000 T 0024
09066000 T 0024
09067000 T 0024
09068000 T 0024
09069000 T 0024
09070000 T 0024
09071000 T 0024
09072000 T 0024
09073000 T 0024
09074000 T 0024

```

OCT0564132154331566, OCT0575160607420123, OCT0606414751324150,
OCT0621012014361120, OCT0631214417455344, OCT0641457523370635,
OCT0651773450267005, OCT0662372362344606, OCT0673071057035747,
OCT0703707272645341, OCT0714671151416632, OCT0726047403722400,
OCT0737461304707100, OCT0751137556607072, OCT0761367512350710,
OCT0771665435043072,

COMMENT THIS IS THE FILL FOR THE SECOND ROW OF INFO!
THE FIRST ITEMS ARE STREAM RESERVED WORDS,
THEN ORDINARY RESERVED WORDS,
THEN INTRINSIC FUNCTIONS;

FILL INFO[1,*] WITH

OCT0670000600000002, "2SI000",
OCT0700001040000002, "2DI000",
OCT0710001460000002, "2CI000",
OCT0720001630000002, "5TALLY",
OCT0730000530000002, "2DS000",
OCT0740000150000002, "4SKIP0",
OCT0750001620000002, "4JUMP0",
OCT0760000740000002, "2DB000",
OCT0770000500000002, "2SB000",
OCT1010000730000002, "2SC000",
OCT1020001160000002, "3LOC00",
OCT1030001170000002, "2DC000",
OCT1040001430000002, "5LOCAL",
OCT1050000340000002, "3LIT00",
OCT1060001036400002, "3SET00",
OCT1060001066500002, "5RESET",
OCT1060001020500002, "3WDS00",
OCT1060001357700002, "3CHR00",
OCT1060001057300002, "3ADD00",
OCT1060001617200002, "3SUB00",
OCT1060000727600002, "3ZON00",
OCT1060000417500002, "3NUM00",
OCT1060000766700002, "3OCT00",
OCT1060000176600002, "3DEC00",
OCT1004000260000003, "6TOGGL", "E0000000",
OCT0130311060000002, "3ABS00",
OCT1360441030000002, "3AND00",
OCT0500000170000002, "5ARRAY",
OCT0660000000000002, "5BEGIN",
OCT0500000040000003, "7BOOLE", "AN000000",
OCT1070000000000003, "7COMME", "NT000000",
OCT0500000230000003, "6DEFIN", "E0000000",
OCT1410446000000002, "3DIV00",
OCT0550000000000002, "2DO000",
OCT0520000000000003, "6DOUBL", "E0000000",
OCT0570000000000002, "4ELSE0",
OCT0600000000000002, "3END00",
OCT1340442030000002, "3EQV00",
OCT0410000000000002, "5FALSE",
OCT0130310030000002, "4FLAG0",
OCT0530000000000002, "3FOR00",
OCT1100000000000003, "7FORWA", "RD",
OCT0640000000000002, "2GO000",
OCT0130316060320002, "4HUNT0",

X256
X258
X260
X262
X264
X266
X268
X270
X272
X274
X276
X278
X280
X282
X284
X286
X288
X290
X292
X294
X296
X298
X300
X302
X304
X307
X309
X311
X313
X315
X318
X321
X324
X326
X328
X331
X333
X335
X337
X339
X341
X343
X346
X348

09075000 T 0024
09076000 T 0024
09077000 T 0024
09078000 T 0024
09079000 T 0024
09080000 T 0024
70 IS 115 LONG, NEXT SEG 69
09081000 T 0024
09082000 T 0024
09083000 T 0024
09084000 T 0024
09085000 T 0024
09086000 T 0025
START OF SEGMENT ***** 71
09087000 T 0026
09088000 T 0026
09089000 T 0026
09090000 T 0026
09091000 T 0026
09092000 T 0026
09093000 T 0026
09094000 T 0026
09095000 T 0026
09096000 T 0026
09097000 T 0026
09098000 T 0026
09099000 T 0026
09100000 T 0026
09101000 T 0026
09102000 T 0026
09103000 T 0026
09104000 T 0026
09105000 T 0026
09106000 T 0026
09107000 T 0026
09108000 T 0026
09109000 T 0026
09110000 T 0026
09110001 T 0026
09112000 T 0026
09112100 T 0026
09112200 T 0026
09112300 T 0026
09112400 T 0026
09112500 T 0026
09112600 T 0026
09112700 T 0026
09112800 T 0026
09112900 T 0026
09113000 T 0026
09113100 T 0026
09113200 T 0026
09113300 T 0026
09113400 T 0026
09113500 T 0026
09113600 T 0026
09113700 T 0026

```
OCT2025, COMMENT DUP 1
OCT0000, COMMENT LITC 0
OCT0425, COMMENT NEG 1
OCT1025, COMMENT XCH 1
OCT0155, COMMENT DIA 1
OCT0161, COMMENT DIB 1
OCT0165, COMMENT TRB 1
```

%350
%352
%354
%357
%359
%362
%364
%367
%369
%371
%373
%375
%377
%379
%381
%384
%387

%396
%398
%401
%404
%407
%409
%411
%413
%415
%417
%419
%421
%423
%425
%427
%429
%431
%433
%435
%437
%439
%441
%443
%445
%447
%449
%451
%453
%455
%457
%459
%461
%463

09113800	T	0026
09113900	T	0026
09114000	T	0026
09114100	T	0026
09114200	T	0026
09114300	T	0026
09114400	T	0026
09114500	T	0026
09114600	T	0026
09114700	T	0026
09114800	T	0026
09114900	T	0026
09115000	T	0026
09115100	T	0026
09115200	T	0026
09115300	T	0026
09115400	T	0026
09115500	T	0026
09115600	T	0026
09115700	T	0026
09115800	T	0026
09115900	T	0026
09116000	T	0026
09116100	T	0026
09116200	T	0026
09116300	T	0026
09116400	T	0026
09116500	T	0026
09116600	T	0026
09116700	T	0026
09116800	T	0026
09116900	T	0026
09117000	T	0026
09117100	T	0026
09117200	T	0026
09117300	T	0026
09117400	T	0026
09117500	T	0026
09117600	T	0026
09117700	T	0026
09117800	T	0026
09117900	T	0026
09118000	T	0026
09118100	T	0026
09118200	T	0026
09118300	T	0026
09118400	T	0026
09118500	T	0026
09118600	T	0026
09118700	T	0026
09118800	T	0026
09118900	T	0026
09119000	T	0026
09119050	T	0026
09119100	T	0026
09119200	T	0026
09119300	T	0026

OCT1310250420000002,	"3INB00",	%465	09119400 T	0026	
OCT1310100420000002,	"3INI00",	%467	09119500 T	0026	
OCT1400440300000002,	"3INX00",	%469	09119600 T	0026	
OCT1310244220000002,	"3IOR00",	%471	09119700 T	0026	
OCT1310250220000002,	"3IP100",	%473	09119800 T	0026	
OCT1310250420000002,	"3IP200",	%475	09119900 T	0026	
OCT1310145060000002,	"3IPS00",	%477	09120000 T	0026	
OCT1310410240000002,	"3ISD00",	%479	09120100 T	0026	
OCT1310450440000002,	"3ISN00",	%481	09120200 T	0026	
OCT1310100420000002,	"3ITI00",	%483	09120300 T	0026	
OCT1310450250000002,	"3LEQ00",	%485	09120400 T	0026	
OCT1310505300000002,	"3LLL00",	%487	09120500 T	0026	
OCT1310441030000002,	"3LND00",	%489	09120600 T	0026	
OCT1310300230000002,	"3LNG00",	%491	09120700 T	0026	
OCT1310304040000002,	"3LOD00",	%493	09120800 T	0026	
OCT1310440430000002,	"3LOR00",	%495	09120900 T	0026	
OCT1310442030000002,	"3LQV00",	%497	09121000 T	0026	
OCT1310450450000002,	"3LSS00",	%499	09121100 T	0026	
OCT1310101100000002,	"3MKS00",	%501	09121200 T	0026	
OCT1310441000000002,	"3MUL00",	%503	09121300 T	0026	
OCT1310441050000002,	"3NEQ00",	%505	09121400 T	0026	
OCT1310100130000002,	"3NOP00",	%507	09121500 T	0026	
OCT0650006550000003,	"6SCOP0", "N.....";	%509	09121600 T	0026	
FILL INFO[2,*] WITH			71 IS	256 LONG,	NEXT SEG 69
OCT1310300000020004,	"3RDF00",	%512	09121650 T	0026	
			09121700 T	0027	
			START OF SEGMENT	*****	72
OCT0000,	COMMENT LITC 0;		09121800 T	0028	
OCT2141,	COMMENT FXS ;		09121900 T	0028	
OCT1310300000020004,	"3RDS00",	%516	09122000 T	0028	
OCT0004,	COMMENT LITC 1;		09122100 T	0028	
OCT2141,	COMMENT FXS ;		09122200 T	0028	
OCT1310456000000002,	"3RDV00",	%520	09122300 T	0028	
OCT1310304030000002,	"3RFB00",	%522	09122400 T	0028	
OCT1310240470000002,	"3RND00",	%524	09122500 T	0028	
OCT1310145060000002,	"3RRR00",	%526	09122600 T	0028	
OCT1310311060000002,	"3RSB00",	%528	09122700 T	0028	
OCT1310242470000002,	"3RSP00",	%530	09122800 T	0028	
OCT1310141020000002,	"3RTM00",	%532	09122900 T	0028	
OCT1310240470000002,	"3RTN00",	%534	09123000 T	0028	
OCT1310141020000002,	"3RTR00",	%536	09123100 T	0028	
OCT1310242470000002,	"3RTS00",	%538	09123200 T	0028	
OCT1310310030000002,	"3SFB00",	%540	09123300 T	0028	
OCT1310442040000002,	"3SND00",	%542	09123400 T	0028	
OCT1310301060000002,	"3SSB00",	%544	09123500 T	0028	
OCT1310316060000002,	"3SSF00",	%546	09123600 T	0028	
OCT1310301060000002,	"3SSN00",	%548	09123700 T	0028	
OCT1310311060000002,	"3SSP00",	%550	09123800 T	0028	
OCT1310401040000002,	"3STD00",	%552	09123900 T	0028	
OCT1310240000020004,	"3STF00",	%554	09124000 T	0028	
OCT0010,	COMMENT LITC 2;		09124100 T	0028	
OCT2141,	COMMENT FXS ;		09124200 T	0028	
OCT1310442040000002,	"3STN00",	%558	09124300 T	0028	
OCT1310240000020004,	"3STS00",	%560	09124400 T	0028	
OCT0014,	COMMENT LITC 3;		09124500 T	0028	
OCT2141,	COMMENT FXS ;		09124600 T	0028	
OCT1310440600000002,	"3SUB00",	%564	09124700 T	0028	

OCT1310344060000002, "3TFB00",
 OCT1270440650000002, "3TFR00",
 OCT1310155060000002, "3TI000",
 OCT1310344060000002, "3TOP00",
 OCT1270440650000002, "3TRB00",
 OCT1300300000000002, "3VFI00",
 OCT1310502050000002, "3XCH00",
 OCT1310101070000002, "3XIT00",
 OCT1310105020000002, "3ZIP00",
 OCT1310105020000002, "3ZP100",
 OCT1270500750000002, "3CFE00",
 OCT1270500750000002, "3FCE00",
 OCT1270500710000002, "3CFL00",
 OCT1270500710000002, "3FCL00",
 OCT1310440210000002, "3DLA00",
 OCT1310440210000002, "3ADL00",
 OCT1310440610000002, "3DLS00",
 OCT1310440610000002, "3SDL00",
 OCT1310441010000002, "3DLM00",
 OCT1310441010000002, "3MDL00",
 OCT1310442010000002, "3DLDO0",
 OCT1310442010000002, "3DDL00",
 OCT0460000000000002, "1P0000",
 OCT0360002000020002, "1M0000",
 OCT1310240000020004, "3PRL00",
 OCT0111, COMMENT PRL;
 OCT0055, COMMENT NOP;
 OCT0650006610000003, "7SCOP0", "FF.....",
 OCT0030000000040003, "2LB000", "[# ",
 OCT0030000000040003, "2RB000", "]# ",
 OCT0030000000040003, "3GTR00", "># ",
 OCT0030000000040003, "3GEQ00", "2# ",
 OCT0030000000040003, "3EQL00", "# ",
 OCT0030000000040003, "3NEQ00", "x# ",
 OCT0030000000040003, "3LEQ00", "≤# ",
 OCT0030000000040003, "3LSS00", "<# ",
 OCT0030000000040003, "5TIMES", "x# ",
 OCT1310117530000002, "3SCIO0",
 OCT1310117540000002, "3SAN00",
 OCT1310157730000000, "3SCS00",

%566
 %568
 %570
 %572
 %574
 %576
 %578
 %580
 %582
 %584
 %586
 %588
 %590
 %592
 %594
 %596
 %598
 %600
 %602
 %604
 %606
 %608
 %610
 %612
 %614

 %618
 %621
 %624
 %627
 %630
 %633
 %636
 %639
 %642
 %645
 %648
 %650
 %652

09124800 T 0028
 09124900 T 0028
 09125000 T 0028
 09125050 T 0028
 09125100 T 0028
 09125200 T 0028
 09125300 T 0028
 09125400 T 0028
 09125500 T 0028
 09125600 T 0028
 09125700 T 0028
 09125800 T 0028
 09125900 T 0028
 09126000 T 0028
 09126100 T 0028
 09126200 T 0028
 09126300 T 0028
 09126400 T 0028
 09126500 T 0028
 09126600 T 0028
 09126700 T 0028
 09126800 T 0028
 09126900 T 0028
 09127000 T 0028
 09127100 T 0028
 09127200 T 0028
 09127300 T 0028
 09127400 T 0028
 09127500 T 0028
 09127600 T 0028
 09127700 T 0028
 09127800 T 0028
 09127900 T 0028
 09128000 T 0028
 09128100 T 0028
 09128200 T 0028
 09128300 T 0028
 09128400 T 0028
 09128500 T 0028
 09128600 T 0028
 09128700 T 0028
 09128800 T 0028
 09128900 T 0028
 09129000 T 0028
 09129100 T 0028
 09129200 T 0028
 09129300 T 0028
 09129400 T 0028
 09129500 T 0028
 09129600 T 0028
 09129700 T 0028
 09129800 T 0028
 09129900 T 0028
 09130000 T 0028
 09130100 T 0028
 09130200 T 0028
 09130300 T 0028

0; % END OF INFO FILL.

```
FOR GT2*256 STEP GT1, LINK WHILE NOT BOOLEAN(GT1, FORMAL) DO
  PUT((GT1+TAKE(GT2))&GT2[35:35:13], GT2);
FOR GT1*GT2 STEP GT2, LINK WHILE GT2, LINK#0 DO
  PUT((GT2+TAKE(GT1))&STACKHEAD[GT3+TAKE(GT1+1), [12:36]
    MOD 125][35:35:13], STACKHEAD[GT3]+GT1);
```

COMMENT THIS IS THE FILL FOR THE SPECIAL CHARACTORS;

FILL SPECIAL[*] WITH

OCT1200000000200000,	COMMENT	#;	OCT0000000000100000,	COMMENT	@;
OCT0000000000000000,			OCT1160000000120000,	COMMENT	!;
OCT1370440450002763,	COMMENT	>;	OCT1370440250002662,	COMMENT	2;
OCT1400440200000000,	COMMENT	+;	OCT0000000000000000,		
OCT1220000000060000,	COMMENT	;	OCT1210000000000000,	COMMENT	[;
OCT1250000000000000,	COMMENT	&;	OCT0450000000000000,	COMMENT	(;
OCT1370450450003571,	COMMENT	<;	OCT1330401040000000,	COMMENT	+;
OCT1410441000000000,	COMMENT	x;	OCT0000000000000000,		
OCT0000000000040000,	COMMENT	\$;	OCT0470000000000000,	COMMENT	*;
OCT1400440600000000,	COMMENT	=;	OCT1240000000160000,	COMMENT	);
OCT0620000000000000,	COMMENT	.,;	OCT1370450250003470,	COMMENT	;
OCT0000000000000000,			OCT1410442000000000,	COMMENT	/;
OCT1170000000000000,	COMMENT	,;	OCT0000000000020000,	COMMENT	%;

09130400	T	0028
09130500	T	0028
09130600	T	0028
09130700	T	0028
09130800	T	0028
09130900	T	0028
09131000	T	0028
09131100	T	0028
09131200	T	0028
09131300	T	0028
09131400	T	0028
09131500	T	0028
09131600	T	0028
09131700	T	0028
09131800	T	0028
09131900	T	0028
09132000	T	0028
09132100	T	0028
09132200	T	0028
09132300	T	0028
09132400	T	0028
09132500	T	0028
09132600	T	0028
09132700	T	0028
09132800	T	0028
09132900	T	0028
09133000	T	0028
09133100	T	0028
09133200	T	0028
09133300	T	0028
09133400	T	0028
09133450	T	0028
09133500	T	0028
09133600	T	0028
09133700	T	0028

72 IS 143 LONG, NEXT SEG 69

09133800	T	0028
09133900	T	0034
09134000	T	0040
09134100	T	0045
09134200	T	0049
09197000	T	0053
09198000	T	0053
09199000	T	0053

START OF SEGMENT ***** 73

09200000	T	0054
09201000	T	0054
09202000	T	0054
09203000	T	0054
09204000	T	0054
09205000	T	0054
09206000	T	0054
09207000	T	0054
09208000	T	0054
09209000	T	0054
09210000	T	0054
09211000	T	0054

OCT1370441050002561, COMMENT #; OCT1370451050002460, COMMENT =;
 OCT1230000000000000, COMMENT]; OCT0000000000140000, COMMENT ";
 0,0;

FILL MACRO[*] WITH
 OCT0131,

	COMMENT SFS A	00	;
OCT0116,	COMMENT SFD A	01	;
OCT0000,	COMMENT SYNTAX ERROR02		;
OCT0140,	COMMENT INC A	03	;
OCT0130,	COMMENT SRS A	04	;
OCT0117,	COMMENT SRD A	05	;
OCT0000,	COMMENT SYNTAX ERROR06		;
OCT0000,	COMMENT SYNTAX ERROR07		;
OCT00310143,	COMMENT CRF A, SFS 008		;
OCT00160143,	COMMENT CRF A, SFD 009		;
OCT00470143,	COMMENT CRF A, JFW 0 10		;
OCT00400143,	COMMENT CRF A, INC 011		;
OCT00300143,	COMMENT CRF A, SRS 012		;
OCT00170143,	COMMENT CRF A, SRD 013		;
OCT0000,	COMMENT SYNTAX ERROR14		;
OCT0000,	COMMENT SYNTAX ERROR15		;
OCT0153,	COMMENT RSA A	16	;
OCT0104,	COMMENT RDA A	17	;
OCT0150,	COMMENT RCA A	18	;
OCT004201430042,	COMMENT SEC 0, CRF A, SEC 0 19		;
OCT0122,	COMMENT SES A	20	;
OCT0106,	COMMENT SED A	21	;
OCT0000,	COMMENT SYNTAX ERROR22		;
OCT0000,	COMMENT SYNTAX ERROR23		;
OCT0056,	COMMENT TSA 0	24	;
OCT0000,	COMMENT SYNTAX ERROR25		;
OCT0000,	COMMENT SYNTAX ERROR26		;
OCT0000,	COMMENT SYNTAX ERROR27		;
OCT0000,	COMMENT SYNTAX ERROR28		;
OCT0007,	COMMENT TDA 0	29	;
OCT0000,	COMMENT SYNTAX ERROR30		;
OCT0000,	COMMENT SYNTAX ERROR31		;
OCT0115,	COMMENT SSA A	32	;
OCT0114,	COMMENT SDA A	33	;
OCT0154,	COMMENT SCA A	34	;
OCT0141,	COMMENT STC A	35	;

FILL OPTIONS[*] WITH "5CHECK",0, % 0,1

"6DEBUG",0,	% 2,3
"4DECK0",0,	% 4,5
"6FORMA",0,	% 6,7
"9INTRI",0,	% 8,9
"5LISTA",0,	% 10,11
"4LIST0",0,	% 12,13
"5LISTP",0,	% 14,15
"3MCP00",0,	% 16,17
"4TAPE0",0,	% 18,19
"4NEST0",0,	% 20,21
"3NEW00",0,	% 22,23
"7NEWIN",0,	% 24,25

09212000 T	0054	
09213000 T	0054	
09214000 T	0054	
73 IS	32 LONG, NEXT SEG	69
09215000 T	0054	
09216000 T	0055	
START OF SEGMENT	*****	74
09217000 T	0056	
09218000 T	0056	
09219000 T	0056	
09220000 T	0056	
09221000 T	0056	
09222000 T	0056	
09223000 T	0056	
09224000 T	0056	
09225000 T	0056	
09226000 T	0056	
09227000 T	0056	
09228000 T	0056	
09229000 T	0056	
09230000 T	0056	
09231000 T	0056	
09232000 T	0056	
09233000 T	0056	
09234000 T	0056	
09235000 T	0056	
09236000 T	0056	
09237000 T	0056	
09238000 T	0056	
09239000 T	0056	
09240000 T	0056	
09241000 T	0056	
09242000 T	0056	
09243000 T	0056	
09244000 T	0056	
09245000 T	0056	
09246000 T	0056	
09247000 T	0056	
09248000 T	0056	
09249000 T	0056	
09250000 T	0056	
09251000 T	0056	
74 IS	36 LONG, NEXT SEG	69
09251208 T	0056	
START OF SEGMENT	*****	75
09251212 T	0058	
09251214 T	0058	
09251216 T	0058	
09251218 T	0058	
09251220 T	0058	
09251224 T	0058	
09251228 T	0058	
09251230 T	0058	
09251232 T	0058	
09251234 T	0058	
09251236 T	0058	
09251240 T	0058	


```

"40MITO",0,      % 26,27
"130000",0,      % 28,29
"3PRT00",0,      % 30,31
"5PUNCH",0,      % 32,33
"5PURGE",0,      % 34,35
"4SEGS0",0,      % 36,37
"3SEQ00",0,      % 38,39
"6SEQR",0,       % 40,41
"6SINGL",0,      % 42,43
"5STUFF",0,      % 44,45
"4VUID0",0,      % 46,47
"5VUIDT",0,      % 48,49

```

0;

```

DO UNTIL STEPI = BEGINV;
GT1 += " ";
INTOG + INTOG AND TRUE; %
DISKADR + IF INTOG THEN INTRINSICADR ELSE 2;
MDESC(GT1,INFO[LASTSEQROW, LASTSEQUENCE]);
MDESC(GT1,INFO[LASTSEQROW, LASTSEQUENCE-1]);
MDESC(GT1,INFO[LASTSEQROW, LASTSEQUENCE-2]);
STMT;
LOCK(STUFF);
CLOSE(CARD, RELEASE);
IF LASTUSED # 1 THEN CLOSE(TAPE, RELEASE);
IF NEWTOG THEN LOCK(NEWTAPE, *);
IF T*((L+3)DIV 4) + CORADR > 4080 THEN FLAG(040);
IF NOT NOHEADING THEN % PRINT THESE THINGS IF ANY
    BEGIN % LISTING HAS BEEN DONE.

```

STREAM PROCEDURE PAN(T, FIEL, NER, LSQ); VALUE NER, T;

PRT(642) = *SEGMENT DESCRIPTOR*

PRT(643) = PAN

```

BEGIN DI + FIEL; 44(DS+2LIT " ");
SI + LSQ; DS + WDS; SI + FIEL; DS + 3 WDS;
DI + FIEL; DS + 28 LIT "NUMBER OF ERRORS DETECTED = ";
SI + LOC NER; DS + 3 DEC; DS + 22 LIT " COMPILATION TIME = ";
SI + LOC T; DS + 4 DEC; DS + 9 LIT " SECONDS."; END;

```

STREAM PROCEDURE PEN(FIL, PRTSIZ, BASE, CORE, DISK);

PRT(644) = PEN

```

VALUE PRTSIZ, BASE, CORE, DISK;
BEGIN DI + FIL; DS + 9 LIT "PRT SIZE="; SI + LOC PRTSIZ;
DS + 3 DEC; DS + 14 LIT " BASE ADDRESS=";
SI + LOC BASE; DS + 4 DEC; DS + 10 LIT " CORE REQ=";
SI + LOC CORE; DS + 4 DEC; DS + 10 LIT " DISK REQ=";
SI + LOC DISK; DS + 5 DEC; DS + 61 LIT " ";
END PEN;

```

STREAM PROCEDURE FINALAX(LINE, N, SEQ); VALUE N;

PRT(645) = FINALAX

```

09251244 T 0058
09251248 T 0058
09251252 T 0058
09251256 T 0058
09251260 T 0058
09251264 T 0058
09251268 T 0058
09251272 T 0058
09251276 T 0058
09251278 T 0058
09251280 T 0058
09251284 T 0058
09251288 T 0058

```

75 IS 51 LONG, NEXT SEG 69

```

09252000 T 0058
09253000 T 0060
09253050 T 0061
09253100 T 0064
09253500 T 0067
09254000 T 0069
09255000 T 0072
09275000 T 0074
09281000 T 0075
09281500 T 0076
09282000 T 0078
09282500 T 0081
09282600 T 0084
09362000 T 0088
09363000 T 0089
09364000 T 0089

```

START OF SEGMENT ***** 76

```

09365000 T 0000
09366000 T 0001
09367000 T 0002
09368000 T 0006
09369000 T 0009

```

09370000 T 0012

```

09371000 T 0012
09372000 T 0012
09373000 T 0014
09374000 T 0016
09375000 T 0018
09376000 T 0020
09377000 T 0028

```

09378000 T 0029


```

        BEGIN DI ← LINE; 15(DS ← 8 LIT " ");
          DI ← LINE; DS ← 31 LIT "NUMBER OF ACCIDENTAL ENTRIES = ";
          SI ← LOC N; DS ← 3 DEC; DI ← DI+8;
          SI ← SEQ; SI ← SI-16; DS ← 8 CHR;
        END;

        IF AXNUM ≠ 0 THEN
          BEGIN
            FINALAX(LIN[0],AXNUM,INFO[LASTSEQROW, LASTSEQUENCE]);
            WRITELINE;
          END;
        SCRAM := (TIME(1)-TIME1)/60;
        PAN(SCRAM,LIN[0],ERRORCOUNT,INFO[LASTSEQROW, LASTSEQUENCE-1]);
        WRITELINE;
        PEN(LIN[0],PRTIMAX,T:=(L+3)DIV 4,T:=CORADR+T,
          ((T+29)DIV 30+DISKADR)×30);
        WRITELINE;
        LOCK(LINE,RELEASE); END;

PRT(646) = *SEGMENT DESCRIPTOR*

        IF ERRORCOUNT ≠ 0 THEN I+0/0 ELSE
          BEGIN
            ARRAY SAVINFO[0:31,0:255];
PRT(647) = *SEGMENT DESCRIPTOR*

STACK(F+2) = SAVINFO
          INFO[0:200,0:255]; % FOR LARGE MCP'S.
STACK(F+3) = INFO
          INTEGER SAVNDX, NONSAVNDX, N;

STACK(F+4) = SAVNDX
STACK(F+5) = NONSAVNDX
STACK(F+6) = N
          INTEGER Q, J, K, M;

STACK(F+7) = Q
STACK(F+10) = J
STACK(F+11) = K
STACK(F+12) = M
          BOOLEAN TSSTOG; REAL T;

STACK(F+13) = TSSTOG
STACK(F+14) = T
          REAL PROCEDURE PUSHER(GRINCH, GOT, XMAS); VALUE XMAS; REAL XMAS;

PRT(650) = PUSHER
          ARRAY GOT[0]; ARRAY GRINCH [0,0];

          BEGIN
            REAL WHO, WHAT;

STACK(F+3) = WHO
STACK(F+4) = WHAT
          %
          DEFINE LINKR = [32:8]#;

          IF WHO=XMAS.LINKC ≤ 225 THEN
            BEGIN
              MOVE(30, GRINCH[XMAS.LINKR, WHO], GOT[0]);

```

```

09379000 T 0029
09380000 T 0031
09381000 T 0035
09382000 T 0036
09383000 T 0037

```

```

09384000 T 0037
09384050 T 0038
09384100 T 0039
09384500 T 0041
09384600 T 0052
09385000 T 0052
09386000 T 0054
09386500 T 0057
09387000 T 0057
09388000 T 0067
09389000 T 0071
09389500 T 0074
09390000 T 0084

```

```

76 IS 87 LONG, NEXT SEG 69
09391000 T 0092
09392000 T 0094
09392300 T 0095

```

START OF SEGMENT ***** 77

```

09392500 T 0003
09393000 T 0005

```

```

09393010 T 0005

```

```

09393020 T 0005

```

```

09393050 T 0005

```

```

09393060 T 0005
09393070 T 0005
09393080 T 0005

```

START OF SEGMENT ***** 78

```

09393090 T 0000
09393100 T 0000
09393110 T 0000
09393120 T 0001
09393130 T 0002

```

```

        PUSHER:=XMAS + 30;
      END
    ELSE BEGIN
      MOVE(WHAT:=256-WHO,GRINCH[XMAS,LINKR,WHU],GOT[0]);
      XMAS:=XMAS + WHAT;
      MOVE(WHO:=30-WHAT, GRINCH[XMAS,LINKR,0], GOT[WHAT]);
      PUSHER:=XMAS + WHO;
    END;
  END PUSHER;

```

```

09393140 T 0005
09393150 T 0006
09393160 T 0006
09393170 T 0007
09393180 T 0011
09393190 T 0012
09393200 T 0016
09393220 T 0017
09393230 T 0017
78 IS 20 LONG, NEXT SEG 77

```

```

PRT(651) = PUSHEE

```

```

PROCEDURE PUSHEE(GRINCH,N,B,Y); VALUE N,B,Y; REAL N,B,Y;

```

```

  ARRAY GRINCH[0,0];
  BEGIN
    REAL I,J,X;

```

```

STACK(F+2) = I
STACK(F+3) = J
STACK(F+4) = X

```

```

  DEFINE LINKR = [32:8]#;
  J:=Y;
  I:=B + N;
  WHILE B < I DO
    BEGIN
      IF Y:=B, LINKC ≤ 225 THEN
        BEGIN
          MOVE(30, CODE(J), GRINCH[B, LINKR, Y]);
          J:=J + 30;
          B:=B + 30;
        END
      ELSE BEGIN
        MOVE(X:=256-Y, CODE(J), GRINCH[B, LINKR, Y]);
        B:=B + X;
        J:=J + X;
        MOVE(Y:=30-X, CODE(J), GRINCH[B, LINKR, 0]);
        B:=B + Y;
        J:=J + Y;
      END;
    END;
  END PUSHEE;

```

```

09393240 T 0005
09393250 T 0005
09393260 T 0005
09393270 T 0005
START OF SEGMENT ***** 79

```

```

09393280 T 0000
09393290 T 0000
09393300 T 0000
09393310 T 0002
09393320 T 0003
09393330 T 0003
09393340 T 0005
09393350 T 0005
09393360 T 0011
09393370 T 0012
09393380 T 0013
09393390 T 0013
09393400 T 0014
09393410 T 0020
09393420 T 0021
09393430 T 0023
09393440 T 0029
09393450 T 0030
09393460 T 0032
09393470 T 0032
09393480 T 0032
79 IS 35 LONG, NEXT SEG 77

```

```

PRT(652) = FIXHDR

```

```

STREAM PROCEDURE FIXHDR(F,N); VALUE N;

```

```

  BEGIN SI:=F; SI:=SI-24; DI:=LOC F; DS:=WDS;
    SI:=F; 14(SI+SI+8); DI:=LOC F; DS:=WDS;
    DI:=F; DI:=DI+38; SI:=LOC N;
    SI:=SI+7; DS:=CHR;
  END FIXHDR;

```

```

09393700 T 0005
09393710 T 0005
09393720 T 0007
09393730 T 0008
09393740 T 0009
09393750 T 0009

```

```

        LABEL EOF;
    IF NOT INTOG THEN
    BEGIN
        L+((L+3)DIV 4; COMMENT L+NUM. OF WORDS IN OUTER BLOCK;
        FILL SAVINFO[0,*] WITH
            OCT77000000000000015;

            OCT0253010477527705;
            UCT0051000000000000;
            OCT0441070001000062;

        W ← -1;
        PUSHEE(SAVINFO,L,4,5);
        SAVNDX:=L;
    END;
    REWIND(CODISK);
    DO BEGIN IF REED=0 THEN GO TO EOF;
        N+FETCH(MKABS(CODE(0)))=1;
        IF BOOLEAN(FETCH(MKABS(CODE(1)))) THEN
    BEGIN
        PUSHEE(SAVINFO,N,SAVNDX,1);
        SAVNDX:=SAVNDX +N;
    END ELSE BEGIN
        IF DECKTOG THEN
            STACKHEAD[Q+Q+1] + 1024×NONSAVNDX+N;
            PUSHEE(INFO,N,NONSAVNDX,1);
            NONSAVNDX:=((NONSAVNDX + N + 29)DIV 30)×30;
        END;
        END UNTIL FALSE;
    EOF: N+((SAVNDX+29) DIV 30; COMMENT NUMBER OF DISK SEGMENTS
        OCCUPIED BY SAVE PROCEDURES AND ARRAYS;
    IF INTOG AND NOT DECKTOG THEN
    BEGIN % INTRINSIC FUNCTION OPTION
        FOR J:=USEROPINX STEP 2 UNTIL OPARSIZE DO % IS TIMESHARING SET
        IF OPTIONS[J] = "TIMES" THEN
        BEGIN TSSTUG:=BOOLEAN(OPTIONS[J+1]); J:=OPARSIZE END;
        I ← PRTBASE + 1; J ← 0;
        DO IF GT1 + PRT[I] ≠ 0 THEN
        BEGIN
            J ← J+1;
            SAVINFO[J,LINKR,J,LINKC] ←
                O&GT1[8:8:10]
                &GT1[33:18:15];
        END UNTIL I:=I +1 ≥ PRTIMAX;
        SAVINFO[0,0] ← J; % # OF INTRINSICS
        SAVNDX ← MAXINTRINSIC;
    END ELSE BEGIN
        I+PRTBASE; DO IF GT1+PRT[I]≠0 THEN
        BEGIN IF GT1.[1:5]≠LDES THEN
        BEGIN IF (GT1+GT1&(GT1.[33:15]+L)[33:33:15]),[6:2]×3 THEN
            GT1+GT1&(GT1.[18:15]+N)[18:33:15];
        END;
        MDESC(GT1,SAVINFO[I,LINKR,I,LINKC]);
        END ELSE SAVINFO[I,LINKR,I,LINKC]:=0 UNTIL I:=I+1≥PRTIMAX;
        MDESC(O&1[2:47:1],SAVINFO[0,PRTBASE=1]);
    
```

```

09394000 T 0010
09394100 T 0010
09394200 T 0011
09395000 T 0011
09395100 T 0013
09395200 T 0014
START OF SEGMENT ***** 80
09395300 T 0015
09395400 T 0015
09395500 T 0015
80 IS 4 LONG, NEXT SEG 77
09395700 T 0015
09396000 T 0016
09397000 T 0018
09397100 T 0019
09398000 T 0019
09399000 T 0021
09400000 T 0022
09401000 T 0028
09402000 T 0033
09402100 T 0034
09403000 T 0036
09404000 T 0037
09405000 T 0038
09405500 T 0038
09406000 T 0042
09407000 T 0044
09408000 T 0047
09412000 T 0047
09413000 T 0048
09414000 T 0050
09414010 T 0050
09414020 T 0052
09414022 T 0053
09414024 T 0055
09414026 T 0056
09414030 T 0061
09414040 T 0063
09414050 T 0065
09414060 T 0065
09414070 T 0066
09414080 T 0069
09414090 T 0070
09414100 T 0071
09414110 T 0073
09414120 T 0075
09414130 T 0076
09415000 T 0078
09415500 T 0081
09416000 T 0082
09417000 T 0086
09417500 T 0089
09418000 T 0089
09419000 T 0092
09419100 T 0097

```

```

        SAVNDX ← 30 × N;
END;
    I ← 0; J ← -1;

    IF NOT DECKTOG THEN
    BEGIN
        DO
        BEGIN
            I:=PUSHER(SAVINFO,ELBAT,I);
            J:=J + 1;
            WRITE(DISK,30,ELBAT[*]);
        END UNTIL I ≥ SAVNDX;
        I:=0;
        WHILE I < NONSAVNDX DO
        BEGIN
            I:=PUSHER(INFO,ELBAT,I);
            J:=J + 1;
            WRITE(DISK,30,ELBAT[*]);
        END;
        N←IF INTOG THEN IF TSSTOG THEN
            TSSINTYPE ELSE DCINTYPE ELSE MCPTYPE;
        FIXHDR(DISK,N);
        LOCK(DISK,*);
    END ELSE
    BEGIN ELBAT[0]←0; I←16;
        DO BEGIN MOVE(8,SAVINFO[I,LINKR,I,LINKC],ELBAT[1]);
            ELBAT[9]←B2D(I+96)&1[11:47:1]&(I+96)[23:35:1];
            WRITE(DECK,10,ELBAT[*]);
        END UNTIL I+I+82SAVNDX;
        FILL ELBAT[*] WITH 0,

            OCT7500000000000012,
            OCT0004535530611765,
            OCT7006000404210435,
            OCT7700000000000015,
            OCT0253010477527705,
            OCT0051000004410046,
            OCT0441070001000062,
            OCT0040413100000000,
            OCT0001000000000101;

        WRITE(DECK,10,ELBAT[*]);
        ELBAT[0] ←0&REAL(DECKTOG)[1:19:17];
        FOR I ← 0 STEP 1 UNTIL Q DO
            BEGIN K ← STACKHEAD[I],[23:15];
                M ← STACKHEAD[I],[38:10];
                FOR J ← 0 STEP 8 UNTIL M DO BEGIN
                    MOVE(8,INFO[(J+K),LINKR,(J+K),LINKC],
                        ELBAT [1]);
                    ELBAT[9] ← B2D(J)&"310"[1:31:17];
                    WRITE(DECK,10,ELBAT[*]) END;
                END;
            END END END PROGRAM;

```

PRT(653) = *SEGMENT DESCRIPTOR*

09420000	T	0101	
09420010	T	0102	
09420020	T	0102	
09420100	T	0104	
09421000	T	0104	
09421500	T	0105	
09422000	T	0106	
09423000	T	0106	
09424000	T	0106	
09425000	T	0109	
09425900	T	0110	
09426000	T	0114	
09427000	T	0115	
09427100	T	0116	
09427200	T	0118	
09427500	T	0118	
09428000	T	0121	
09429000	T	0122	
09430000	T	0126	
09430050	T	0127	
09430060	T	0128	
09430075	T	0131	
09430100	T	0132	
09431000	T	0134	
09432000	T	0134	
09433000	T	0137	
09434000	T	0140	
09435000	T	0145	
09436000	T	0149	
09437000	T	0151	
START OF SEGMENT ***** 81			
09438000	T	0153	
09439000	T	0153	
09440000	T	0153	
09441000	T	0153	
09442000	T	0153	
09443000	T	0153	
09444000	T	0153	
09445000	T	0153	
09446000	T	0153	
81 IS	10 LONG,	NEXT SEG	77
09447000	T	0153	
09447010	T	0157	
09447020	T	0160	
09447030	T	0162	
09447040	T	0163	
09447050	T	0165	
09447060	T	0166	
09447070	T	0169	
09447080	T	0170	
09447090	T	0173	
09447100	T	0179	
09448000	T	0182	
77 IS 186 LONG, NEXT SEG 69			
69 IS 101 LONG, NEXT SEG 3			

COMMENT THIS SECTION CONTAINS GENERATORS USED BY THE BLOCK ROUTINE;	10000000 T 0468	
PROCEDURE DEFINEGEN(MACRO,J); VALUE MACRO,J; BOOLEAN MACRO; REAL J;	10228000 T 0468	
PRT(654) = DEFINEGEN		
BEGIN	10229000 T 0468	
OWN INTEGER CHARCOUNT, REMCOUNT;	10230000 T 0468	
	START OF SEGMENT *****	82
PRT(655) = CHARCOUNT		
PRT(656) = REMCOUNT		
COMMENT CHARCOUNT CONTAINS NUMBER OFCHARACTORS OF THE DEFINE THAT WE	10231000 T 0000	
HAVE PUT INTO INFO. REMCOUNT CONTAINS NUMBER OF CHARACT-	10232000 T 0000	
ORS REMAINING IN THIS ROW OF INFO;	10233000 T 0000	
PROCEDURE PUTOGETHER(CHAR); REAL CHAR;	10234000 T 0000	
PRT(657) = PUTOGETHER		
BEGIN	10235000 T 0000	
STREAM PROCEDURE PACKINFO(INFO,ISKIP,COUNT,ASKIP,ACCUM);	10236000 T 0000	
	START OF SEGMENT *****	83
PRT(660) = PACKINFO		
VALUE ISKIP,COUNT,ASKIP;	10237000 T 0000	
BEGIN DI = INFO; DI = DI+ISKIP;	10238000 T 0000	
SI = ACCUM; SI = SI+ASKIP; SI = SI+3;	10239000 T 0000	
DS = COUNT CHR END PACKINFO;	10240000 T 0001	
INTEGER COUNT,SKIPCOUNT;	10241000 T 0002	
STACK(F+2) = COUNT		
STACK(F+3) = SKIPCOUNT		
IF (COUNT + CHAR.[12:6]) + CHARCOUNT > 2047	10242000 T 0002	
THEN BEGIN FLAG(142); TB1= TRUE END	10243000 T 0004	
ELSE BEGIN	10244000 T 0007	
IF COUNT > REMCOUNT	10245000 T 0009	
THEN BEGIN	10246000 T 0009	
SKIPCOUNT = COUNT-(COUNT+REMCOUNT);	10247000 T 0010	
REMCOUNT = 2047 END	10248000 T 0012	
ELSE REMCOUNT = REMCOUNT-COUNT;	10249000 T 0012	
GT1 = CHARCOUNT DIV 8 + NEXTINFO;	10250000 T 0016	
PACKINFO(INFO[GT1.LINKR,GT1.LINKC],CHARCOUNT.[45:3],	10251000 T 0018	
COUNT,0,CHAR);	10252000 T 0021	
IF SKIPCOUNT ≠ 0 THEN	10253000 T 0022	
PACKINFO(INFO[NEXTINFO.LINKR+1,0],0,SKIPCOUNT,	10254000 T 0023	
COUNT,CHAR);	10255000 T 0026	
CHARCOUNT = CHARCOUNT+SKIPCOUNT+COUNT END	10256000 T 0027	
END PUTOGETHER;	10257000 T 0029	
	83 IS 32 LONG, NEXT SEG	82
STREAM PROCEDURE SCAN(D,S,Q,N,J); VALUE J,N,Q;	10257100 T 0000	
PRT(661) = SCAN		
BEGIN DI=D;DI=DI+1;SI=S;SI=SI+3;	10257200 T 0000	
IF N SC=DC THEN	10257300 T 0001	
IF SC>"0" THEN	10257400 T 0001	
BEGIN DI=LOC J; DI=DI+7;	10257500 T 0002	

IF SC\$DC THEN	10257600 T	0003
BEGIN J+SI;DI+J;SI+LOC Q;SI+SI+6;DS+CHR;	10257700 T	0004
DI+S;DI+DI+2;DS+CHR;	10257800 T	0006
END END END;	10257900 T	0006

STACK(F+2) = LASTRESULT	INTEGER LASTRESULT;	10258000 T	0007
STACK(F+3) = K	REAL K,N,ELCLASS;	10258100 T	0007
STACK(F+4) = N			
STACK(F+5) = ELCLASS			
	DEFINE I=NXTELBT#;	10258200 T	0007
	LABEL FINAL,PACKIN;	10258300 T	0007
	LABEL BACK,SKSC,EXIT;	10259000 T	0007
	TB1= FALSE;	10260000 T	0007
	CHARCOUNT=(NEXTINFO-LASTINFO)*8;	10261000 T	0007
	DEFINETR + 1; LASTRESULT + 2;	10262000 T	0009
	REMCOUNT + (256 - NEXTINFO MOD 256) * 8;	10263000 T	0011
	NEXTINFO=LASTINFO;	10263100 T	0013
	IF J#0 THEN N+TAKE(LASTINFO+1),[12:6];	10263110 T	0014
	K+0;	10263200 T	0017
BACK:	STOPDEFINE+TRUE;	10263300 T	0018
	ELCLASS=TABLE(NXTELBT);	10263400 T	0019
SKSC:	NXTELBT+NXTELBT=1;	10263500 T	0021
	IF MACRO THEN	10263600 T	0022
	BEGIN IF ELCLASS=CUMMA THEN	10263700 T	0022
	IF K#0 THEN	10263800 T	0023
FINAL:	BEGIN PUTTOGETHER("1#0000"); GO TO EXIT END	10263900 T	0025
	ELSE GO PACKIN;	10264000 T	0029
	IF ELCLASS=LEFTPAREN OR ELCLASS=LFTBRKET THEN	10264100 T	0029
	BEGIN K+K+1; GO TO PACKIN END;	10264200 T	0030
	IF ELCLASS=RTTPAREN OR ELCLASS=RTBRKET THEN	10264300 T	0033
	IF K+K-1<0 THEN GO FINAL ELSE GO PACKIN;	10264400 T	0034
	IF ELCLASS=SEMICOLON THEN	10264410 T	0038
	BEGIN FLAG(142); GO TO FINAL END ELSE GO PACKIN	10264420 T	0038
	END;	10264500 T	0040
	IF J#0 THEN	10264600 T	0040
	IF ACCUM[1],[12:6]=1=N THEN	10264700 T	0041
	SCAN(INFO[LASTINFO, LINKR ,LASTINFO, LINKC],	10264800 T	0043
	ACCUM[1],N+770,N,J);	10264900 T	0046
PACKIN:		10264910 T	0048
	IF RESULT = 4	10265000 T	0049
	THEN BEGIN	10266000 T	0049
COMMENT	INSERT " MARKS = 2130706432 IS DECIMAL FOR 1"0000;	10267000 T	0050
	PUTTOGETHER(2130706432);	10268000 T	0050
	PUTTOGETHER(ACCUM[1]);	10269000 T	0051
	PUTTOGETHER(2130706432) END	10270000 T	0052
	ELSE BEGIN	10271000 T	0052
	IF BOOLEAN(RESULT) AND BOOLEAN(LASTRESULT)	10272000 T	0055
	THEN PUTTOGETHER("1 0000"); COMMENT INSERT BLANK;	10273000 T	0055
	PUTTOGETHER(ACCUM[1]) END;	10274000 T	0057
	IF TB1 THEN GO TO EXIT;	10275000 T	0058
	LASTRESULT + RESULT;	10276000 T	0059
	IF MACRO THEN GO BACK;	10276500 T	0059

```

        IF ELCLASS=DECLARATORS AND ELBAT[I].ADDRESS = DEFINEV
        THEN BEGIN DEFINECTR + DEFINECTR+1; GO BACK END;
        IF ELCLASS # CROSSHATCH THEN GO BACK;
        IF DEFINECTR # 1
        THEN BEGIN STOPDEFINE + TRUE;
                IF ELCLASS=TABLE(I)#COMMA THEN
                DEFINECTR+DEFINECTR-1; GO SKSC END;
EXIT:   DEFINECTR+ 0;
        NEXTINFO +(CHARCOUNT+7) DIV 8+NEXTINFO;
END     DEFINEGEN;

```

```

10277000 T 0060
10278000 T 0062
10279000 T 0067
10280000 T 0068
10281000 T 0068
10282000 T 0070
10283000 T 0072
10284000 T 0074
10285000 T 0075
10286000 T 0078
82 IS 81 LONG, NEXT SEG 3

```

```

PROCEDURE OBLSTMT;
BEGIN
  REAL S,T;

```

```

12002000 T 0468
12003000 T 0468
12004000 T 0468
START OF SEGMENT ***** 84

```

```

STACK(F+2) = S
STACK(F+3) = T

```

```

LABEL L1,L2,L3,EXIT;
S+0;
IF STEPI#LEFTPAREN THEN ERR(281)
ELSE
L1: BEGIN
  IF STEPI=COMMA THEN
  BEGIN
    DPTOG+TRUE;
    IF STEPI=ADOP THEN STEPIT;
    EMITNUM(NLO);
    EMITNUM(IF ELBAT[I-1].ADDRESS #SUB THEN #NHI ELSE NHI);
    DPTOG+FALSE;
    STEPIT;
    GO TO L2;
  END;
  IF TABLE(I+1)=COMMA THEN
  BEGIN
    IF ELCLASS=ADOP OR ELCLASS=MULOP THEN
    BEGIN
      EMITN(ELBAT[I].ADDRESS+1);
      IF S+S=150 THEN FLAG(282); STEPIT;
      GO TO L3;
    END;
    IF ELCLASS=ASSIGNOP THEN
    BEGIN
      IF S#1 THEN FLAG(283); S+0; STEPIT;
      DO
      BEGIN
        IF ELCLASS #COMMA THEN BEGIN ERR(284);GO EXIT END;
        STEPIT;
        IF ELCLASS$INTID AND ELCLASS$REALID THEN
        BEGIN EMITN(ELBAT[I].ADDRESS); STEPIT END
          ELSE VARIABLE(FL);
        EMITN(STD) END UNTIL S+S+1#2 ;
        IF ELCLASS#RTPAREN THEN ERR(285) ELSE STEPIT;
        GO TO EXIT;
      END;
    END;
  END;

```

```

12005000 T 0000
12006000 T 0000
12007000 T 0000
12008000 T 0002
12009000 T 0003
12010000 T 0004
12011000 T 0005
12012000 T 0005
12013000 T 0006
12014000 T 0008
12015000 T 0009
12016000 T 0013
12017000 T 0014
12018000 T 0014
12019000 T 0015
12020000 T 0015
12021000 T 0016
12022000 T 0017
12023000 T 0019
12024000 T 0019
12025000 T 0021
12026000 T 0025
12027000 T 0025
12028000 T 0025
12029000 T 0026
12030000 T 0026
12031000 T 0030
12032000 T 0030
12033000 T 0030
12034000 T 0032
12035000 T 0033
12036000 T 0034
12037000 T 0037
12038000 T 0038
12039000 T 0041
12040000 T 0044

```

```

        END;
    IF ELCLASS<INTID AND ELCLASS>BODID THEN
    BEGIN
        CHECKER(T+ELBAT[I]);
        STEPIT;STEPIT;
        AEXP;
        EMITV(T,ADDRESS);
        GO TO L2;
    END;
    END
    ;
    AEXP;
    IF ELCLASS#COMMA THEN BEGIN ERR(284);GO EXIT
                                END;
        STEPIT; AEXP; EMITO(XCH);
L2:    S+S+1;
L3:    IF ELCLASS#COMMA THEN BEGIN ERR(284);GO TO EXIT END;
        GO TO L1;
EXIT:END
    END DBLSTMT;

```

```

12041000 T 0045
12042000 T 0045
12043000 T 0046
12044000 T 0047
12045000 T 0048
12046000 T 0049
12047000 T 0050
12048000 T 0051
12049000 T 0052
12050000 T 0052
12051000 T 0052
12052000 T 0052
12053000 T 0055
12054000 T 0055
12055000 T 0056
12056000 T 0058
12057000 T 0061
12058000 T 0062
12059000 T 0062
84 IS 65 LONG, NEXT SEG 3

```

```

REAL PROCEDURE FIXDEFINEINFO(T); VALUE T; REAL T;
    BEGIN REAL K,S,P,J,EL;

```

```

12101000 T 0468
12102000 T 0468
START OF SEGMENT ***** 85

```

```

STACK(F+3) = K
STACK(F+4) = S
STACK(F+5) = P
STACK(F+6) = J
STACK(F+7) = EL

```

```

PRT(662) = SET

```

```

STREAM PROCEDURE SET(S,D,K,E); VALUE K,E;
    BEGIN SI+S;SI+SI+1;DI+D;DI+DI+3;DS+K CHR;
        SI+LOC E; SI+SI+6; DS+2 CHR;
    END;

```

```

12103000 T 0000
12104000 T 0000
12105000 T 0001
12106000 T 0002

```

```

MACROID+TRUE;
P+(FIXDEFINEINFO+T).ADDRESS;
K+COUNT;
S+SCRAM;
STREAMTOG+TRUE & STREAMTOG[1:3:45] ;
STOPDEFINE+TRUE;
EL+TABLE(NXTELBT);
NXTELBT+NXTELBT-1;
IF EL#LEFTPAREN AND EL#LFTBRKET THEN
    FLAG(141)
ELSE DO BEGIN J+J+1;
    SET(INFO[T,LINKR,T,LINKC],ACCUM[1],K,64*J+12);
    ACCUM[1].[12:6]+K+2;
    ACCUM[0]+0;
    ACCUM[0].CLASS+DEFINEDID;
    COUNT+K+2;

```

```

12107000 T 0002
12108000 T 0003
12109000 T 0005
12110000 T 0006
12110100 T 0007
12111000 T 0008
12112000 T 0009
12113000 T 0010
12114000 T 0012
12115000 T 0013
12116000 T 0014
12117000 T 0017
12118000 T 0022
12119000 T 0025
12120000 T 0026
12121000 T 0028

```



```

        SCRAM+ACCUM[1] MOD 125;
        E;
        DEFINE GEN(TRUE,0);
        END UNTIL EL+ELBAT[NXTTELBT],CLASS#COMMA;
        IF EL#RTPAREN AND EL#RTBRKET OR J#P THEN FLAG(141);
        MACROID#FALSE;
        STREAMTOG#STREAMTOG,[1:45] ;
    END;

```

```

12122000 T 0030
12123000 T 0031
12124000 T 0032
12125000 T 0033
12126000 T 0035
12127000 T 0039
12127100 T 0040
12128000 T 0042
85 IS 46 LONG, NEXT SEG 3

```

```

PROCEDURE SCATTERELBAT;
    BEGIN
        REAL T;

STACK(F+2) = T

        T ← ELBAT[I];
        KLASSE ← T.CLASS;
        FORMAL ← BOOLEAN(T.FORMAL);
        VONF ← BOOLEAN(T.VU);
        LEVEL ← T.LVL;
        ADDRSE ← T.ADDRESS;
        INCR ← T.INCR;
        LINK ← T.LINK;
    END SCATTERELBAT;

```

```

13197000 T 0468
13198000 T 0468
13199000 T 0468
START OF SEGMENT ***** 86
13200000 T 0000
13201000 T 0001
13202000 T 0002
13203000 T 0003
13204000 T 0004
13205000 T 0006
13206000 T 0007
13207000 T 0008
13208000 T 0009
86 IS 12 LONG, NEXT SEG 3

```

```

PROCEDURE CHKSQB;
PRT(663) = CHKSQB
    IF GTA1[J+J-1]#0 THEN FLAG(23);

```

```

13209000 T 0468
13210000 T 0468

```

```

DEFINE
    ADDC#532480#,
    SUBC#1581056#,
    EMITSTORE#EMITPAIR#;
PROCEDURE PURGE(STOPPER);
    VALUE STOPPER;
    REAL STOPPER;
    BEGIN
        INTEGER POINTER;

STACK(F+2) = POINTER
        LABEL RECOV; DEFINE ELCLASS = KLASSE#;
        REAL J,N,OCR,TL,ADD;

STACK(F+3) = J
STACK(F+4) = N
STACK(F+5) = OCR
STACK(F+6) = TL
STACK(F+7) = ADD

```

```

13211000 T 0472
13212000 T 0472
13213000 T 0472
13214000 T 0472
13215000 T 0472
13216000 T 0472
13217000 T 0472
13218000 T 0472
13219000 T 0472
START OF SEGMENT ***** 87
13220000 T 0000
13221000 T 0000

```

```

    POINTER←LASTINFO;
    WHILE POINTER ≥ STOPPER
    DO
        BEGIN
            IF ELCLASS←(GT1+TAKE(POINTER)).CLASS=NONLITNO
            THEN BEGIN
                NCII←NCII+1;
                EMITNUM(TAKE(POINTER+1));
                EMITSTORE(MAXSTACK,STD);
                MAXSTACK←(G+MAXSTACK)+1;
                J←L; L←GT1.LINK;
                DO
                    BEGIN
                        GT4←GET(L);
                        EMITV(G);
                    END
                UNTIL (L←GT4)≠4095;
                L←J;
                POINTER←POINTER+GT1.INCR;
            END
            ELSE
                BEGIN
                    IF NOT BOOLEAN(GT1.FORMAL)
                    THEN BEGIN
                        IF ELCLASS = LABELID
                        THEN BEGIN
                            ADD ← GT1.ADDRESS;
                            IF NOT BOOLEAN(OCR←TAKE(GIT(POINTER))).[1:1]
                            THEN IF OCR.[36:12] ≠ 0 OR ADD ≠ 0
                                THEN BEGIN GT1 ← 160; GO TO RECOV END;
                            IF ADD ≠ 0 THEN
                                PROGDESCBLDR(ADD,TRUE,OCR.[36:10],LDES) END
                        ELSE IF FALSE
                        THEN BEGIN
                            IF TAKE(POINTER+1) < 0
                            THEN BEGIN GT1 ← 162; GO TO RECOV END;
                            OCR ← (J ← TAKE(GIT(POINTER))).[24:12];
                            N ← GET( (J+J.[36:12])+4 ); TL ← L;
                            IF ADD ← GT1.ADDRESS ≠ 0
                            THEN BEGIN
                                IF OCR ≠ 0
                                THEN BEGIN L←OCR+2; CALLSWITCH(POINTER); EMITV(BFW);END;
                                    L←J+1; EMITL(15); EMITV(RTS);
                                    FOR J ← 4 STEP 4 UNTIL N
                                    DO BEGIN
                                        EMITL(GNAT(GET(L)×4096+GET(L+1)));
                                        EMITV(RTS) END END
                                    ELSE BEGIN
                                        L ← J+13;
                                        FOR J ← 4 STEP 4 UNTIL N
                                        DO BEGIN
                                            GT1 ← GET(L)×4096+GET(L+1);
                                            GOGEN(GT1,BFW) END;END;
                                        L ← TL END
                                ELSE IF ELCLASS ≥ PROCID AND ELCLASS ≤ INTPROCID
                                THEN IF TAKE(POINTER+1) < 0

```

```

13222000 T 0000
13223000 T 0000
13224000 T 0001
13225000 T 0002
13226000 T 0002
13227000 T 0004
13228000 T 0005
13229000 T 0006
13230000 T 0008
13231000 T 0009
13232000 T 0011
13233000 T 0013
13234000 T 0014
13235000 T 0014
13236000 T 0015
13237000 T 0015
13238000 T 0016
13239000 T 0017
13240000 T 0018
13241000 T 0018
13242000 T 0020
13243000 T 0020
13244000 T 0022
13245000 T 0022
13246000 T 0023
13247000 T 0023
13248000 T 0024
13249000 T 0026
13250000 T 0028
13251000 T 0030
13252000 T 0033
13252500 T 0033
13253000 T 0036
13254000 T 0037
13255000 T 0037
13256000 T 0038
13257000 T 0041
13258000 T 0043
13259000 T 0047
13260000 T 0048
13261000 T 0049
13262000 T 0049
13263000 T 0049
13264000 T 0053
13265000 T 0056
13266000 T 0056
13267000 T 0057
13268000 T 0060
13269000 T 0063
13270000 T 0066
13271000 T 0067
13272000 T 0067
13273000 T 0068
13274000 T 0071
13277000 T 0074
13278000 T 0075
13279000 T 0078

```

```

                THEN BEGIN GT1 ← 161;
RECOV:          MOVE(9,INFO[POINTER,LINKR,POINTER,LINKC],ACCUM);
                Q ← ACCUM[1]; FLAG(GT1); ERRORTOG ← TRUE END
                END;
                GT2←TAKE(POINTER+1);
                GT3←GT2,PURPT;
STACKHEAD[(0&GT2[12:12:36])MOD 125]←TAKE(POINTER).LINK;
                POINTER←POINTER-GT3
                END
                END ;
LASTINFO←POINTER;
NEXTINFO←STOPPER
END;

```

```

13280000 T 0080
13281000 T 0082
13282000 T 0086
13283000 T 0089
13284000 T 0089
13285000 T 0090
13286000 T 0092
13287000 T 0095
13288000 T 0096
13289000 T 0097
13290000 T 0097
13291000 T 0098
13292000 T 0098
87 IS 103 LONG, NEXT SEG 3

```

```

PROCEDURE E;
COMMENT
    E IS THE PROCEDURE WHICH PLACES AN ENTRY IN INFO AND
    HOOKS IT INTO STACKHEAD. THE PREVIOUS STACKHEAD LINK
    IS SAVED IN THE LINK OF THE ELBAT WORD IN THE NEW ENTRY
    E PREVENTS AN ENTRY FROM OVERFLOWING A ROW, STARTING AT THE
    BEGINNING OF THE NEXT ROW IF NECESSARY
BEGIN
    REAL WORDCOUNT,RINX;
STACK(F+2) = WORDCOUNT
STACK(F+3) = RINX
    IF RINX+(NEXTINFO+WORDCOUNT+(COUNT+18)DIV 8 )>.LINKR ≠
        NEXTINFO,LINKR
    THEN BEGIN PUT(125&(RINX×256+NEXTINFO)[27:40:8],NEXTINFO);
        NEXTINFO←256×RINX END;
        IF SPECTOG THEN
            IF NOT MACROID THEN
                UNHOOK;
        ACCUM[0],INCR←WORDCOUNT;
        IF NOT INLINETOGL OR MACROID THEN BEGIN
            ACCUM[0].LINK ←STACKHEAD[SCRAM];STACKHEAD[SCRAM]←NEXTINFO;
            END;
        ACCUM[1],PURPT←NEXTINFO←LASTINFO;
MOVE(WORDCOUNT,ACCUM,INFO[NEXTINFO.LINKR,NEXTINFO.LINKC]);
        LASTINFO←NEXTINFO;
        NEXTINFO←NEXTINFO+WORDCOUNT
    END;

```

```

13293000 T 0472
13294000 T 0472
13295000 T 0472
13296000 T 0472
13297000 T 0472
13298000 T 0472
13299000 T 0472
13300000 T 0472
13301000 T 0472
START OF SEGMENT ***** 88
13302000 T 0000
13303000 T 0003
13304000 T 0003
13305000 T 0007
13305100 T 0009
13305200 T 0009
13305300 T 0010
13306000 T 0011
13307000 T 0011
13307500 T 0013
13308000 T 0015
13308500 T 0019
13309000 T 0019
13310000 T 0022
13311000 T 0025
13312000 T 0026
13313000 T 0026
88 IS 30 LONG, NEXT SEG 3

```

```

PROCEDURE ENTRY(TYPE);
    VALUE TYPE;
    REAL TYPE;
COMMENT
    ENTRY ASSUMES THAT I IS POINTING AT AN IDENTIFIER WHICH

```

```

13314000 T 0472
13315000 T 0472
13316000 T 0472
13317000 T 0472
13318000 T 0472

```

```

        IS BEING DECLARED AND MAKES UP THE ELBAT ENTRY FOR IT
        ACCORD TO TYPE      .IF THE ENTRY IS AN ARRAY AND NOT
        A SPECIFICATION THEN A DESCRIPTOR IS PLACED IN THE STACK
        FOR THE UPCOMING COMMUNICATE TO GET STORAGE FOR THE ARRAY(S) ;
BEGIN
    J+0;I+I-1;
    DO
        BEGIN
            STOPDEFINE +TRUE; STEPIT; SCATTERELBAT;
            IF FORMALF+SPECTOG
            THEN
                BEGIN
                    IF ELCLASS#SECRET
                    THEN FLAG(002);
                    BUP+BUP+1
                ;
                KLASSF+TYPE;MAKEUPACCUM;E;J+J+1;
                END
            ELSE
                BEGIN
                    IF ELCLASS>IDMAX
                    THEN IF ELCLASS= POLISHV THEN ELCLASS+TYPE ELSE FLAG(3);
                    IF LEVELF=LEVEL
                    THEN FLAG(001);
                    VONF+P2;
                    FORMALF+PTOG;
                    KLASSF+TYPE; MAKEUPACCUM;E; J+J+1;
                    IF ((FORMALF+PTOG) OR(STREAMTOG AND NOT STOPGSP)) AND NOT P2
                    THEN ADDR$F+PJ+PJ+1
                    ELSE IF STOPGSP
                    THEN ADDR$F+0
                        ELSE ADDR$F:=GETSPACE(P2, LASTINFO+1);
                    PUT(TAKE(LASTINFO)& ADDR$F[16:37:11],LASTINFO);
                END END
            UNTIL STEPIT#CUMMA OR STOPENTRY; GTA1[0]+J
        END;

```

```

PROCEDURE UNHOOK;
COMMENT
    UNHOOK ASSUMES THAT THE WORD IN ELBAT[I] POINTS TO A PSUEDO ENTRY
    FOR A PARAMETER, ITS JOB IS TO UNHOOK THAT FALSE ENTRY SO THAT
    E WILL WORK AS NORMAL.
BEGIN
    REAL LINKT, A, LINKP;

```

```

STACK(F+2) = LINKT
STACK(F+3) = A
STACK(F+4) = LINKP

```

LABEL L;

```

        LINKT+STACKHEAD[SCRAM]      ; LINKP+ELBAT[I],LINK;
        IF LINKT=LINKP THEN STACKHEAD[SCRAM]+TAKE(LINKT),LINK
        ELSE
        L: IF A+TAKE(LINKT),LINK=LINKP
        THEN PUT((TAKE(LINKT))&(TAKE(A))[35:35:13],LINKT)

```

```

13319000 T 0472
13320000 T 0472
13321000 T 0472
13322000 T 0472
13323000 T 0472
13324000 T 0472
13325000 T 0475
13326000 T 0475
13327000 T 0475
13328000 T 0476
13329000 T 0476
13330000 T 0477
13331000 T 0478
13332000 T 0478
13333000 T 0480
13333500 T 0480
13334000 T 0484
13335000 T 0484
13336000 T 0484
13337000 T 0484
13338000 T 0485
13339000 T 0489
13340000 T 0489
13341000 T 0491
13341100 T 0492
13342000 T 0492
13343000 T 0495
13344000 T 0497
13345000 T 0499
13346000 T 0501
13347000 T 0502
13348000 T 0505
13349000 T 0507
13350000 T 0507
13351000 T 0507
13352000 T 0510

```

```

13353000 T 0511
13354000 T 0511
13355000 T 0511
13356000 T 0511
13357000 T 0511
13358000 T 0511
13359000 T 0511

```

START OF SEGMENT ***** 89

```

13360000 T 0000
13361000 T 0000
13362000 T 0002
13363000 T 0005
13364000 T 0006
13365000 T 0008

```

```

ELSE BEGIN LINKT+A; GO TO L END;
END;

```

```

13366000 T 0012
13367000 T 0014
89 IS 17 LONG, NEXT SEG 3

```

```

PROCEDURE MAKEUPACCUM;
BEGIN
  IF PTOG
    THEN GT1+LEVELF ELSE GT1+LEVEL;
  ACCUM[0]+ ABS(ELBAT[I] & KLASSF[2:41:7] & REAL(FORMALF)[9:47:1]
    & REAL(VONF)[10:47:1] & GT1[11:43:5] & ADURSF[16:37:11]
  )
END;

```

```

13368000 T 0511
13369000 T 0511
13370000 T 0511
13371000 T 0512
13372000 T 0514
13373000 T 0517
13374000 T 0520
13375000 T 0520

```

```

PROCEDURE ARRAE;
PRT(664) = ARRAE
BEGIN
  INTEGER SAVEINFO;
STACK(F+2) = SAVEINFO
  LABEL BETA1;
  TYPEV+REALARRAYID;
  IF T1+GTA1[J+J-1]=0 THEN J+J+1
  ELSE
    IF T1=OWNV THEN
      BEGIN
        P2+TRUE; IF SPECTOG THEN
          FLAG(13)
        END
      ELSE
        TYPEV+REALARRAYID+T1=REALV;
  BETA1: ENTER(TYPEV);
  IF ELCLASS#LFTBRKET THEN FLAG(16);
  IF STEPI#LITNO THEN
    BEGIN
      SAVEINFO+ELBAT[I].ADDRESS;
      IF STEPI#RTBRKET THEN FLAG(53);
      FILLSTMT(SAVEINFO);
    SAVEINFO+1;
    END
  ELSE
    BEGIN IF ELCLASS#ASTRISK THEN FLAG(56);
      SAVEINFO+1;
      WHILE STEPI#RTBRKET DO
        BEGIN IF ELCLASS#COMMA AND
          STEPI#ASTRISK THEN FLAG(56);
          SAVEINFO+SAVEINFO+1
        END; STEPIT;
    END;
  PUT(TAKE(LASTINFO)&SAVEINFO[27:40:8],LASTINFO);
  J + 1 ; GTA1[0] + 0 ;
  IF ELCLASS=COMMA THEN BEGIN STEPIT; GO TO BETA1 END

```

```

13376000 T 0521
13377000 T 0521
13378000 T 0521
START OF SEGMENT ***** 90
13379000 T 0000
13380000 T 0000
13381000 T 0000
13382000 T 0004
13383000 T 0005
13384000 T 0006
13385000 T 0006
13386000 T 0007
13387000 T 0008
13388000 T 0009
13389000 T 0009
13390000 T 0011
13391000 T 0012
13392000 T 0014
13393000 T 0015
13394000 T 0016
13395000 T 0017
13396000 T 0020
13397000 T 0020
13398000 T 0021
13399000 T 0021
13400000 T 0021
13401000 T 0024
13402000 T 0024
13403000 T 0026
13404000 T 0027
13405000 T 0029
13406000 T 0030
13407000 T 0032
13408000 T 0032
13408500 T 0034
13409000 T 0036

```

END ARRAE;

13410000 T 0038
90 IS 41 LONG, NEXT SEG 3

```
PROCEDURE PUTNBUMP(X);  
    VALUE X;  
    REAL X;  
BEGIN  
    INFO[NEXTINFO, LINKR, NEXTINFO, LINKC] ← X;  
    NEXTINFO ← NEXTINFO + 1;  
END ;
```

13589000 T 0521
13590000 T 0521
13591000 T 0521
13592000 T 0521
13593000 T 0521
13594000 T 0525
13595000 T 0525

```
PROCEDURE JUMPCHKX;  
COMMENT THIS PROCEDURE IS CALLED AT THE START OF ANY EXECUTABLE CODE  
        WHICH THE BLOCK MIGHT EMIT. IT DETERMINES WHETHER ANY JUMPS  
        AROUND NONEXECUTABLE CODE MAY BE WAITING AND WHETHER IT  
        IS THE FIRST EXECUTABLE CODE;  
IF NOT SPECTOG THEN  
BEGIN  
    IF AJUMP  
    THEN  
    BEGIN ADJUST;  
        EMITB(BFW, SAVED, L)  
    END ELSE  
        IF FIRSTX ≠ 4095  
        THEN  
        BEGIN  
            ADJUST;  
            FIRSTX ← L;  
        END;  
        AJUMP ← FALSE  
END;
```

13596000 T 0526
13597000 T 0526
13598000 T 0526
13599000 T 0526
13600000 T 0526
13601000 T 0526
13602000 T 0527
13603000 T 0528
13604000 T 0528
13605000 T 0528
13606000 T 0529
13607000 T 0530
13608000 T 0530
13609000 T 0531
13610000 T 0531
13611000 T 0532
13612000 T 0532
13613000 T 0533
13614000 T 0533
13615000 T 0533

```
PROCEDURE JUMPCHKNX;  
COMMENT JUMPCHKNX DETERMINES WHETHER ANY EXECUTABLE CODE HAS BEEN  
        EMITTED AND IF SO WHETHER IT WAS JUST PREVIOUS TO THE  
        NON EXECUTABLE ABOUT TO BE EMITTED. IF BOTH THEN L IS BUMPED  
        AND SAVED FOR A LATER BRANCH;  
IF NOT SPECTOG THEN  
BEGIN  
    IF FIRSTX ≠ 4095  
    THEN  
    BEGIN  
        IF NOT AJUMP  
        THEN  
            SAVED ← BUMPL;  
        AJUMP ← TRUE  
    END; ADJUST  
END;
```

13616000 T 0536
13617000 T 0536
13618000 T 0536
13619000 T 0536
13620000 T 0536
13621000 T 0536
13622000 T 0536
13623000 T 0537
13624000 T 0537
13625000 T 0537
13626000 T 0538
13627000 T 0538
13628000 T 0538
13629000 T 0541
13630000 T 0541
13631000 T 0541

```

PROCEDURE SEGMENTSTART(SAVECODE); VALUE SAVECODE; BOOLEAN SAVECODE;
BEGIN
  STREAM PROCEDURE PRINT(SAVECODE,ADR,FIEL); VALUE SAVECODE,ADR;

PRT(665) = PRINT
  BEGIN
  LABEL L1;
  DI:=FIEL; DS:=8 LIT" ";
  SI:=FIEL; DS:=9 WDS; DI:=DI-3;
  SAVECODE(DS:=38 LIT "START OF SAVE SEGMENT; BASE ADDRESS = ";
  JUMP OUT TO L1);
  DS:=38 LIT " START OF REL SEGMENT; DISK ADDRESS = ";
L1:
  SI:=LOC ADR; DS:=5 DEC;
  END PRINT;

```

```

13632000 T 0544
13632100 T 0544
13633000 T 0544
START OF SEGMENT ***** 91

```

```

13634000 T 0000
13635000 T 0000
13636000 T 0000
13637000 T 0001
13638000 T 0002
13639000 T 0008
13640000 T 0009
13641000 T 0014
13642000 T 0014
13643000 T 0014

```

```

MOVE(1,SAVECODE,CODE(0));
IF SAVECODE AND INTOG AND NOT DECKTOG THEN FLAG(57);
IF LISTER OR SEGSTOG THEN
  BEGIN
  PRINT(SAVECODE,IF SAVECODE THEN CORADR ELSE DISKADR,LIN[*]);
  IF NOHEADING THEN DATIME; WRITELINE;
  END;
END SEGMENTSTART;

```

```

13651000 T 0014
13651100 T 0019
13652000 T 0022
13652500 T 0024
13653000 T 0024
13653500 T 0027
13654000 T 0039
13655000 T 0039
91 IS 40 LONG, NEXT SEG 3

```

```

PROCEDURE SEGMENT(SIZE,FR); VALUE SIZE,FR; INTEGER SIZE,FR;
BEGIN
  STREAM PROCEDURE PRINT(SIZE,FIEL); VALUE SIZE;

PRT(666) = PRINT
  BEGIN
  DI:=FIEL; DS:=8 LIT" ";
  SI:=FIEL; DS:=14 WDS;
  DI:=DI-16; DS:=6 LIT"SIZE=" ";
  SI:=LOC SIZE; DS:=4 DEC; DS:=6 LIT" WORDS"
  END PRINT;

```

```

13657000 T 0544
13660000 T 0544
13661000 T 0544
START OF SEGMENT ***** 92

```

```

13663000 T 0000
13665000 T 0000
13667000 T 0001
13668000 T 0002
13670000 T 0003
13673000 T 0004

```

```

STREAM PROCEDURE DOIT(C,A,I,S,F,W); VALUE C,A,F,W;

PRT(667) = DOIT
  BEGIN LOCAL N;
  DI:=S; DS:=8 LIT" "; SI:=S; DS:=9 WDS;
  DI:=DI-8; SI:=LOC W; DS:=4 DEC;
  SI:=I; SI:=SI+10; DI:=LOC N; DI:=DI+7; DS:=CHR;

```

```

13673100 T 0005
13673150 T 0005
13673200 T 0005
13673250 T 0007
13673300 T 0007

```

```

DI:=S; SI:=LOC F; SI:=SI+7; DS:=CHR; SI:=LOC C;
DS:=3 DEC; DS:=4 DEC; SI:=1; SI:=SI+11; DS:=N CHR;
END DOIT;

```

```

13673350 T 0009
13673400 T 0010
13673450 T 0011

```

```

IF LISTER OR SEGSTOG THEN
  BEGIN
    PRINT(SIZE,LINE[*]);
    IF NOHEADING THEN DATIME; WRITELINE;
  END;
IF STUFFTOG THEN IF FR>0 THEN IF LEVEL>1 THEN
  BEGIN
    KLASSF:=TAKE(PROINFO).CLASS;
    IF FR > 1024 THEN FR:=FR-1024;
    DOIT(KLASSF,FR,INFO[PROINFO.LINKR,PROINFO.LINKC],
      TWXA[0],SAF,SIZE);
    WRITE(STUFF,10,TWXA[*]);
  END;
IF SIZE>SEGSIZEMAX THEN SEGSIZEMAX:=SIZE;
END SEGMENT;

```

```

13674000 T 0012
13674500 T 0013
13675000 T 0013
13676000 T 0015
13677000 T 0026
13677100 T 0026
13677150 T 0029
13677200 T 0030
13677250 T 0032
13677300 T 0034
13677400 T 0037
13677500 T 0039
13677600 T 0043
13678000 T 0043
13681000 T 0045

```

92 IS 47 LUNG, NEXT SEG 3

```

STREAM PROCEDURE MOVECODE(EDOC,TEDOC);
PRT(670) = MOVECODE
  BEGIN LOCAL T1,T2,T3;
    SI+EDOC;T1+SI;
    SI+TEDOC;T2+SI;
    SI+LOC EDOC;
    SI+SI+3;
    DI+LOC T3;
    DI+DI+5;
    SKIP 3 DB;
  15(IF SB THEN DS+ 1 SET ELSE DS+1 RESET;SKIP 1 SB);
    SI+ LOC EDOC;
    DI+ LOC T2;
    DS+ 5 CHR;
    3(IF SB THEN DS+1 SET ELSE DS+1 RESET;SKIP 1 SB);
    DI+T3;
    SI+LOC T2;
    DS+WDS;
    DI+LOC T3;
    DI+DI+5;
    SKIP 3 DB;
    SI+LOC TEDOC;
    SI+SI+3;
  15(IF SB THEN DS+1 SET ELSE DS+ 1 RESET;SKIP 1 SB);
    SI+ LOC TEDOC;
    DI+ LOC T1;
    DS+ 5 CHR;
    3(IF SB THEN DS+1 SET ELSE DS+1 RESET;SKIP 1 SB);
    DI+T3;
    SI+LOC T1;

```

```

13683000 T 0544
13684000 T 0544
13685000 T 0544
13686000 T 0544
13687000 T 0545
13688000 T 0545
13689000 T 0545
13690000 T 0545
13691000 T 0546
13692000 T 0546
13693000 T 0548
13694000 T 0548
13695000 T 0548
13696000 T 0549
13697000 T 0551
13698000 T 0551
13699000 T 0551
13700000 T 0551
13701000 T 0552
13702000 T 0552
13703000 T 0552
13704000 T 0552
13705000 T 0553
13706000 T 0555
13707000 T 0555
13708000 T 0555
13709000 T 0555
13710000 T 0557
13711000 T 0558

```


DS+WDS
END;

13712000 T 0558
13713000 T 0558

```

PROCEDURE ENTER(TYPE);
    VALUE TYPE;
    REAL TYPE;
BEGIN
    G+GTA1[J+J-1];
    IF NOT SPECTOG
    THEN
        BEGIN
            IF NOT P2
            THEN IF P2+(G=OWNV)
            THEN G+GTA1[J+J-1];
            IF NOT P3
            THEN IF P3+(G=SAVEV)
            THEN G+GTA1[J+J-1];
        END;
    IF G#0 THEN FLAG(25) ELSE ENTRY(TYPE)
END;

```

13714000 T 0558
13715000 T 0558
13716000 T 0558
13717000 T 0558
13718000 T 0558
13719000 T 0561
13720000 T 0561
13721000 T 0561
13722000 T 0562
13723000 T 0562
13724000 T 0563
13725000 T 0566
13726000 T 0566
13727000 T 0568
13728000 T 0570
13729000 T 0571
13730000 T 0574

```

PROCEDURE HTTEQAP(GOTSTORAGE,RELAD,STOPPER,PRTAD);
PRT(671) = HTTEQAP
    VALUE GOTSTORAGE,RELAD,STOPPER,PRTAD;
    BOOLEAN GOTSTORAGE;
    REAL RELAD,STOPPER,PRTAD;
BEGIN
    IF FUNCTOG
    THEN
        BEGIN
            EMITV(513);
            EMITO(RTN)
        END
    ELSE
        EMITO(XIT);
    CONSTANTCLEAN;
    PURGE(STOPPER);
    MOVE(1,CODE(0),Z); PROGDESCBLDR(PRTAD,BOOLEAN(Z),(L+3)DIV 4,PDES);
END HTTEQAP;

```

13731000 T 0575
13732000 T 0575
13733000 T 0575
13734000 T 0575
13735000 T 0575
13736000 T 0575
13737000 T 0575
13738000 T 0575
13739000 T 0575
13740000 T 0576
13741000 T 0576
13742000 T 0577
13743000 T 0577
13744000 T 0578
13745000 T 0579
13746000 T 0579
13747000 T 0586

```

PROCEDURE INLINE;
BEGIN
    INTEGER SN,LN,P,LS,J; BOOLEAN MKST;

```

13748000 T 0586
13749000 T 0586
13750000 T 0586

START OF SEGMENT ***** 93

STACK(F+2) = SN
STACK(F+3) = LN
STACK(F+4) = P
STACK(F+5) = LS

STACK(F+6) = J
STACK(F+7) = MKST

STACK(F+10) = FLIPFLOP
STACK(F+11) = PN

```

        BOOLEAN FLIPFLOP;
        INTEGER PN;
        LABEL L1,L2,L3;
        PN+1;
        FLIPFLOP+INLINETO+TRUE;P+0;MKST+FALSE;LS+L;EMIT0(NOP);
        IF STEPI≠LEFTPAREN THEN FLAG(59);
        IF TABLE(I+1)=COLON THEN BEGIN STEPIT;GO TO L2 END;
L1: IF STEPI>IDMAX THEN BEGIN FLAG(465); GO TO L2 END;
        ACCUM[0]+0&P[16:37:11]&LOCLID[2:41:7]&SCRAM[35:35:13];
        E;IF FLIPFLOP THEN BEGIN FLIPFLOP+FALSE;LN+SN+LASTINFO END;
        IF STEPI=COMMA OR ELCLASS=COLON OR ELCLASS=RTTPAREN
        THEN BEGIN I+I+2;STEPIT END
        ELSE IF ELCLASS≠ASSIGNOP THEN FLAG(60) ELSE STEPIT;
        AEXP;
L2: IF ELCLASS=COLON THEN
        BEGIN IF MKST THEN FLAG(99); MKST+TRUE; EMIT0(MKS); P+P+2;
              IF TABLE(I+1)≠RTTPAREN THEN GO TO L1; STEPIT
              ;PN+2;
        END ELSE P+P+1; COUNT UP PARAMS WE HAVE SEEN
        IF ELCLASS=COMMA THEN GO TO L1;
        IF ELCLASS≠RTTPAREN THEN FLAG(61);
        IF NOT MKST THEN
        BEGIN J+L;L+LS;EMIT0(MKS);L+J END;
        IF STEPI ≠ SEMICOLON THEN FLAG(62);
        EMIT0(584);

L3:ELBAT[I]+TAKE(SN);SCATTERELBAT;ADDRSF+P=ADDRSF;
        PUT(ELBAT[I]&ADDRSF[16:37:11]&STACKHEAD[LINKF][33:33:15],SN);
        STACKHEAD[LINKF]+SN; SN+SN+INCRF;
        IF ADDRSF≠PN THEN GO TO L3;
        INLINETO+ FALSE;
        PN+NEXTINFO;
        STREAMTO+TRUE;STREAMWORDS;IF STEPI≠BEGINV THEN STREAMSTMT
        ELSE BEGIN STEPIT;COMPOUNDTAIL END;
        STREAMTO+FALSE;PURGE(PN);STREAMWORDS;PURGE(LN);EMITL(16);

END INLINE;

```

ENTER FORMAL
PARAMS INTO
INFO

13750500	T	0000
13750600	T	0000
13751000	T	0000
13751100	T	0000
13752000	T	0000
13753000	T	0005
13753100	T	0007
13754000	T	0010
13755000	T	0013
13755500	T	0018
13756000	T	0021
13757000	T	0023
13758000	T	0026
13759000	T	0030
13760000	T	0030
13761000	T	0031
13761100	T	0036
13761110	T	0038
13761200	T	0040
13762000	T	0041
13763000	T	0043
13764000	T	0045
13765000	T	0045
13766000	T	0049
13766100	T	0051
13766200	T	0052
13766300	T	0052
13766400	T	0052
13766500	T	0052
13767000	T	0052
13768000	T	0055
13769000	T	0058
13770000	T	0061
13770500	T	0062
13770600	T	0063
13771000	T	0063
13772000	T	0066
13773000	T	0068
13773500	T	0072
13774000	T	0072

93 IS 76 LONG, NEXT SEG 3

```

COMMENT THIS SECTION CONTAINS THE BLOCK ROUTINE ;
PROCEDURE BLOCK(SOP);
        VALUE SOP;
        BOOLEAN SOP;
COMMENT SOP IS TRUE IF THE BLOCK WAS CALLED BY ITSELF THROUGH THE
        PROCEDURE DECLARATION=OTHERWISE IT WAS CALLED BY STATEMENT.
        THE BLOCK ROUTINE IS RESPONSIBLE FOR HANDLING THE BLOCK
        STRUCTURE OF AN ALGOL PROGRAM=SEGMENTING EACH BLOCK,HANDLING

```

14000000	T	0586
14001000	T	0586
14002000	T	0586
14003000	T	0586
14004000	T	0586
14005000	T	0586
14006000	T	0586
14007000	T	0586

ALL DECLARATIONS, DOING NECESSARY BOOKKEEPING REGARDING EACH
BLOCK, AND SUPPLYING THE SCANNER WITH ALL NECESSARY INFORMATION
ABOUT DECLARED IDENTIFIERS.
IT ALSO WRITES EACH SEGMENT ONTO THE PCT;

```
BEGIN
  LABEL OWNERR,SAVERR,BOOLEANDEC,REALDEC,ALPHADEC,INTEGERDEC,
    LABELDEC,DUMPDEC,SUBDEC,OUTDEC,INDEC,MONITORDEC,
    SWITCHDEC,PROCEDUREDEC,ARRAYDEC,NAMEDEC,FILEDEC,
    GOTCHK,
    STREAMERR,DEFINEDEC,CALLSTATEMENT,HF,START;
  SWITCH DECLSW + OWNERR,SAVERR,BOOLEANDEC,REALDEC,INTEGERDEC,ALPHADEC,
    LABELDEC,DUMPDEC,SUBDEC,OUTDEC,INDEC,MONITORDEC,
    SWITCHDEC,PROCEDUREDEC,ARRAYDEC,NAMEDEC,FILEDEC,
    STREAMERR,DEFINEDEC;
  DEFINE NLOCS=10#,LOCBEGIN=PRTI#,
    LBP=[36112]#,
    SPACEITDOWN = BEGIN  WRITE(LINE[DBL]); WRITE(LINE[DBL])  END#;
```

```
    BOOLEAN GOTSTORAGE;
  STACK(F+2) = GOTSTORAGE
    INTEGER PINFOD,BLKAD;
  STACK(F+3) = PINFOD
  STACK(F+4) = BLKAD
```

COMMENT LOCAL TO BLOCK TO SAVE WHERE A PROCEDURE IS ENTERED
IN INFO;

```
    REAL MAXSTACKO,LASTINFOT,RELAD,LO,TSUBLEVEL,STACKCTRO;
  STACK(F+5) = MAXSTACKO
  STACK(F+6) = LASTINFOT
  STACK(F+7) = RELAD
  STACK(F+10) = LO
  STACK(F+11) = TSUBLEVEL
  STACK(F+12) = STACKCTRO
```

```
    INTEGER SGNOO,LOLD,SAVELO,PRTIO,NINFOO;
  STACK(F+13) = SGNOO
  STACK(F+14) = LOLD
  STACK(F+15) = SAVELO
  STACK(F+16) = PRTIO
  STACK(F+17) = NINFOO
```

```
    INTEGER NCIIIO;
  STACK(F+20) = NCIIIO
    INTEGER PROAD ;
  STACK(F+21) = PROAD
```

```
    INTEGER FIRSTXU;
  STACK(F+22) = FIRSTXU
    BOOLEAN FUNCTOGO,AJUMPO;
  STACK(F+23) = FUNCTOGO
  STACK(F+24) = AJUMPO
```

```
    BEGINCTR+BEGINCTR+1;
  IF SOP
```

```
    THEN BEGIN BLKAD+PROAD;
  IF LASTENTRY ≠ 0
    THEN BEGIN GT1+BUMPL;
      CONSTANTCLEAN;
      EMITB(BFW,GT1,L)
    END
```

END

14008000	T	0586	
14009000	T	0586	
14010000	T	0586	
14011000	T	0586	
14012000	T	0586	
14013000	T	0586	
START OF SEGMENT	*****		94
14014000	T	0000	
14015000	T	0000	
14016000	T	0000	
14017000	T	0000	
14018000	T	0000	
14019000	T	0002	
14020000	T	0002	
14021000	T	0002	
14022000	T	0012	
14023000	T	0012	
14023100	T	0012	
14024000	T	0012	
14025000	T	0012	
14026000	T	0012	
14027000	T	0012	
14028000	T	0012	
14029000	T	0012	
14030000	T	0012	
14031000	T	0012	
14032000	T	0012	
14033000	T	0012	
14034000	T	0012	
14035000	T	0012	
14036000	T	0014	
14037000	T	0014	
14038000	T	0015	
14039000	T	0016	
14040000	T	0018	
14041000	T	0019	
14042000	T	0020	
14043000	T	0020	

```
ELSE BEGIN BLKAD:=GETSPACE(TRUE,-6); % SEG. DESCR,
```

```
END;
```

```
    FIRSTXO:=FIRSTX;
    FIRSTX:=0;
    LEVEL:=LEVEL+1;
    LOLO:=L;FUNCTOGO:=FUNCTOG;AJUMPO:=AJUMP;PRTIO:=PRTI;SGNOO:=SGNO;
    SAVELO:=SAVEI;AJUMP:=FALSE; L:=0;NINFOO:=NEXTINFO;
    NCIOO:=NCII;
    NCII:=0;
    STACKCTRO:=STACKCTR;
```

```
    ELBAT[1],CLASS+SEMICOLON;
    START: IF TABLE(1)≠SEMICOLON
```

```
    THEN
        BEGIN
            FLAG(0);
            I:=I-1
```

```
        END;
        GTA1[0]:=J+0;
        IF SPECTOG
```

```
        THEN
            BEGIN
                IF BUP=PJ
                    THEN
                        BEGIN
```

```
                            BEGIN LABEL GETLP;
```

```
PRT(672) = *SEGMENT DESCRIPTOR*
```

```
                IF STREAMTOG THEN F:=0 ELSE
                    F:=FZERO;
                BUP:=LASTINFO;
```

```
                DO
                    BEGIN
                        IF NOT STREAMTOG THEN
```

```
                            BUP:=LASTINFO;
                            GETLP: G:=TAKE(BUP);
                            IF K+G.ADDRESS≠PJ
                                THEN
```

```
                                    BEGIN
                                        IF BUP ≠ BUP:=BUP+TAKE(BUP + 1).PURPT THEN
                                            GO TO GETLP
                                        END;
```

```
TYPEV:=G.CLASS;
```

```
                G.ADDRESS:=F+F+1;
                PUT(G,BUP); G,INCR:=GT1;
                PUT(G,MARK+PJ)
                ;BUP:=BUP+TAKE(BUP+1).PURPT
```

```
            END
            UNTIL PJ+PJ=1=0
        END;
```

```
PRT(673) = *SEGMENT DESCRIPTOR*
```

```
14044000 T 0020
14045000 T 0022
14046000 T 0022
14047000 T 0022
14048000 T 0022
14049000 T 0022
14050000 T 0022
14051000 T 0022
14052000 T 0023
14053000 T 0024
14054000 T 0025
14055000 T 0029
14056000 T 0032
14057000 T 0033
14058000 T 0033
14059000 T 0034
14061000 T 0034
14062000 T 0034
14063000 T 0037
14064000 T 0037
14065000 T 0038
14066000 T 0038
14067000 T 0039
14068000 T 0039
14069000 T 0040
14070000 T 0042
14071000 T 0042
14072000 T 0042
14073000 T 0043
14074000 T 0043
14075000 T 0044
14076000 T 0044
```

```
START OF SEGMENT ***** 95
```

```
14077000 T 0000
14078000 T 0001
14079000 T 0002
14080000 T 0003
14081000 T 0004
14082000 T 0004
14083000 T 0004
14084000 T 0005
14085000 T 0007
14086000 T 0008
14087000 T 0009
14088000 P 0009
14089000 T 0012
14090000 T 0012
14091000 T 0013
14115000 T 0014
14116000 T 0017
14117000 T 0020
14118000 T 0020
14119000 T 0023
14120000 T 0024
14121000 T 0025
```

```

                                SPECTOG=FALSE;
                                GO TO HF
                                END
                                END;
                                STACKCT + 0;
WHILE STEPI=DECLARATORS
DO
BEGIN
    GTA1[J+J+1]=ELBAT[I].ADDRESS;
    STOPDEFINE+ERRORTOG+TRUE;
END;
IF J =0 THEN GO TO CALLSTATEMENT;
P2+P3+FALSE;
GO TO DECLSW[GTA1[J]];
OWNERRIFLAG(20);J+J+1;GO TO REALDEC;
SAVERRIFLAG(21);J+J+1;GO TO REALDEC;
STREAMERR: IF ELCLASS = LEFTPAREN THEN
BEGIN
    I + I - 1;
    GO TO CALLSTATEMENT;
END;
    FLAG(22);
    J + J + 1;
    GO TO PROCEDUREDEC;
REALDEC:P3+TRUE;ENTER(REALID);GO TO START;
ALPHADEC:P3+TRUE;ENTER(ALFAID);GO TO START;
BOOLEANDEC:P3+TRUE;ENTER(BOOID);GO TO START;
INTEGERDEC:P3+TRUE;ENTER(INTID);GO TO START;
MONITORDEC:IF SPECTOG
    THEN BEGIN COMMENT ERROR 463 MEANS THAT A MONITOR
        DECLARATION APPEARS IN THE SPECIFICATION
        PART OF A PROCEDURE;
        FLAG(463);
    END;
    DO UNTIL FALSE;
DUMPDEC:IF SPECTOG
    THEN BEGIN COMMENT ERROR 464 MEANS A DUMP DECLARATION
        APPEARS IN THE SPECIFICATION PART OF A
        PROCEDURE;
        FLAG(464);
    END;
    DO UNTIL FALSE;
ARRAYDEC:ARRAE;GO TO START;
FILEDEC:INDEC:OUTDEC;
GOTCHK:GOTSTORAGE+ NOT SPECTOG OR GOTSTORAGE;GO TO START;
NAMEDEC: IF T1+GTA1[J+J-1]≠ARRAYV THEN J+J+1;
    TYPEV+NAMEID;
    IF T1+GTA1[J+J-1]=0 THEN J+J+1
    ELSE
        IF T1=OWNV
        THEN
            BEGIN
                P2+TRUE; IF SPECTOG THEN
                    FLAG(013)
            END
        ELSE

```

95 IS	27 LONG	NEXT SEG	94
14122000	T	0045	
14123000	T	0045	
14124000	T	0046	
14125000	T	0046	
14125500	T	0046	
14126000	T	0047	
14127000	T	0047	
14128000	T	0048	
14129000	T	0048	
14130000	T	0051	
14131000	T	0052	
14132000	T	0053	
14133000	T	0054	
14134000	T	0055	
14135000	T	0058	
14136000	T	0060	
14137000	T	0063	
14137100	T	0064	
14137200	T	0065	
14137300	T	0066	
14137400	T	0067	
14137500	T	0067	
14137600	T	0067	
14137700	T	0069	
14138000	T	0069	
14139000	T	0072	
14140000	T	0074	
14141000	T	0076	
14142000	T	0078	
14143000	T	0078	
14144000	T	0078	
14145000	T	0078	
14146000	T	0078	
14147000	T	0079	
14148000	T	0079	
14149000	T	0080	
14150000	T	0081	
14151000	T	0081	
14152000	T	0081	
14153000	T	0081	
14154000	T	0082	
14155000	T	0082	
14156000	T	0083	
14158000	T	0085	
14160000	T	0085	
14161000	T	0087	
14161010	T	0091	
14161020	T	0092	
14161030	T	0095	
14161040	T	0096	
14161050	T	0097	
14161060	T	0097	
14161070	T	0098	
14161080	T	0099	
14161090	T	0099	
14161100	T	0100	

```

                                TYPEV+NAMEID+T1=REALV;
                                ENTER(TYPEV); GO TO START;
SUBDEC:
    BEGIN REAL TYPEV,T;
PRT(674) = *SEGMENT DESCRIPTOR*

STACK(F+25) = TYPEV
STACK(F+26) = T
    IF GTA1(J+J-1)=REALV THEN TYPEV+REALSUBID ELSE TYPEV+SUBID;
    STOPGSP+TRUE;
    JUMPCHKNX;ENTRY(TYPEV);IF ELCLASS#SEMICOLON THEN FLAG(57);
    STOPGSP+FALSE;
    STEPIT;
    T+NEXTINFO;
    PUTNBUMP(L); STMT; EMITQ(LFU); IF TYPEV=REALSUBID THEN
        IF GET(L-2)#533 THEN FLAG(58);PUT(TAKE(T)&L[24:36:12],T);
    CONSTANTCLEAN;
    END;
PRT(675) = *SEGMENT DESCRIPTOR*

    GO TO START;

LABELDEC:IF SPECTOG AND FUNCTOG THEN FLAG(24);
    STOPENTRY+STOPGSP+TRUE;
    I+I-1;
    DO
        BEGIN
            STOPDEFINE+TRUE;
            STEPIT;
            ENTRY(LABELID);
            PUTNBUMP(0)
        END
    UNTIL ELCLASS#COMMA;
    STOPENTRY+STOPGSP+FALSE;
    GO TO START;
SWITCHDEC:
    BEGIN
        LABEL START;
PRT(676) = *SEGMENT DESCRIPTOR*

    INTEGER GT1,GT2,GT4,GT5;

```

```

14161110 T 0100
14161120 T 0100
14162000 T 0102
14163000 T 0103
14163500 T 0104

```

START OF SEGMENT ***** 96

```

14164000 T 0000
14164500 T 0004
14165000 T 0005
14166000 T 0008
14166500 T 0009
14167000 T 0010
14168000 T 0013
14168500 T 0019
14169000 T 0019

```

96 IS 21 LONG, NEXT SEG 94

```

14170000 T 0105
14171000 T 0105
14172000 T 0105
14173000 T 0105
14174000 T 0105
14175000 T 0105
14176000 T 0105
14177000 T 0105
14178000 T 0105
14179000 T 0105
14180000 T 0105
14181000 T 0105
14182000 T 0105
14183000 T 0105
14184000 T 0105
14185000 T 0105
14186000 T 0105
14187000 T 0105
14188000 T 0108
14189000 T 0109
14190000 T 0110
14191000 T 0111
14192000 T 0111
14193000 T 0111
14194000 T 0112
14195000 T 0113
14196000 T 0113
14197000 T 0113
14198000 T 0115
14199000 T 0116
14200000 T 0116
14201000 T 0117
14202000 T 0117

```

START OF SEGMENT ***** 97
14203000 T 0000

```

STACK(F+25) = GT1
STACK(F+26) = GT2
STACK(F+27) = GT4
STACK(F+30) = GT5

```

```

STACK(F+31) = TB1
      BOOLEAN TB1;
      STOPENTRY+NOT SPECTOG;STOPGSP+TRUE;
      SCATTERELBAT; GT1+0; TB1+FALSE;
      ENTRY(SWITCHID);
      GT2+NEXTINFO; PUTNBUMP(0);
      DO
      BEGIN
        IF STEPI#LABELID OR ELBAT[I].LVL#LEVEL THEN FLAG(63);
        PUTNBUMP(ELBAT[I]);GT1+GT1+1
      END
      COMMENT
      UNTIL STEPI#COMMA;

      PUT(GT1,GT2);
      STOPENTRY + STOPGSP + FALSE;
      END SWITCHDEC;

```

```

PRT(677) = *SEGMENT DESCRIPTOR*

```

```

      GO TO START;
      DEFINEDEC;
      BEGIN LABEL START;

```

```

PRT(700) = *SEGMENT DESCRIPTOR*

```

```

      REAL J,K;

```

```

STACK(F+25) = J
STACK(F+26) = K

```

```

PRT(701) = PARM

```

```

      BOOLEAN STREAM PROCEDURE PARM(S,D,K,J); VALUE K,J;

```

```

      BEGIN SI+S;SI+SI+2; DI+0;DI+DI+2;
        IF K SC#DC THEN TALLY+1;
        DI+LOC J;DI+DI+7;
        IF SC#DC THEN TALLY+1;
        PARM+TALLY;
      END;

```

```

      STOPENTRY+STOPGSP+TRUE;I+I-1;

```

```

      DO

```

```

      BEGIN

```

```

        STOPDEFINE+TRUE;
        STEPIT; MOVE(9,ACCUM[1],GTA1);
        K+COUNT+1; J+GTA1[0]; ENTRY(DEFINEDID);
        GTA1[0]+J+"100000"; J+0;
        IF ELCLASS=LEFTPAREN OR ELCLASS=LFTBRKET THEN
        BEGIN
          DO BEGIN STOPDEFINE+TRUE;
            STEPIT;
            IF (J+J+1)>9 OR PARM(ACCUM[1],GTA1,K,J) OR
              K>62 THEN BEGIN ERR(141); GO TO START END;
            STOPDEFINE+TRUE;

```

```

14204000 T 0000

```

```

14205000 T 0000

```

```

14206000 T 0001

```

```

14207000 T 0003

```

```

14217000 T 0004

```

```

14218000 T 0006

```

```

14219000 T 0006

```

```

14220000 T 0006

```

```

14221000 T 0010

```

```

14222000 T 0011

```

```

14222500 T 0012

```

```

14223000 T 0012

```

```

14223500 T 0013

```

```

14224000 T 0013

```

```

14251000 T 0014

```

```

14252000 T 0016

```

```

97 IS 17 LONG, NEXT SEG 94

```

```

14253000 T 0118

```

```

14254000 T 0118

```

```

14254050 T 0119

```

```

START OF SEGMENT ***** 98

```

```

14254100 T 0000

```

```

14254200 T 0000

```

```

14254300 T 0000

```

```

14254400 T 0001

```

```

14254500 T 0002

```

```

14254600 T 0002

```

```

14254700 T 0003

```

```

14254800 T 0003

```

```

14255000 T 0004

```

```

14256000 T 0007

```

```

14257000 T 0008

```

```

14258000 T 0008

```

```

14259000 T 0008

```

```

14259010 T 0011

```

```

14259015 T 0014

```

```

14259020 T 0016

```

```

14259030 T 0018

```

```

14259060 T 0018

```

```

14259070 T 0019

```

```

14259080 T 0020

```

```

14259090 T 0024

```

```

14259100 T 0029

```

```

        END UNTIL STEPI#COMMA;
        IF ELCLASS#RTPAREN AND ELCLASS#RTBRKET THEN ERR(141);
        STOPDEFINE#TRUE;
        STEPIT;
        PUT(TAKE(LASTINFO)&J[16:37:11],LASTINFO);
    END;
    IF ELCLASS#RELOP
    THEN
        BEGIN
            FLAG(30);
            I#I-1;
        END;
        MACROID#TRUE;
        DEFINEGEN(FALSE,J);
        MACROID#FALSE;
    END
    UNTIL STEPI#COMMA;
    START: STOPENTRY#STOPGSP#FALSE; END; GO TO START;
PRT(702) = *SEGMENT DESCRIPTOR*

    PROCEDUREDEC;
    BEGIN
        LABEL START,START1;
PRT(703) = *SEGMENT DESCRIPTOR*

        LABEL START2;
        BOOLEAN FWDTOG; COMMENT THIS TOGGLE IS THE FORWARD DEC INDICATOR;
STACK(F+25) = FWDTOG

        IF NOT SPECTOG THEN FUNCTOG#FALSE;
        FWDTOG#FALSE ;
        MAXSTACK# MAXSTACK;
        IF G#GTA1[J+J-1]#STREAMV
        THEN
            BEGIN STREAMTOG#TRUE;
            IF G#GTA1[J+J-1]#0 THEN TYPEV#STRPROCID
            ELSE
                BEGIN
                    IF TYPEV#PROCID +G>INTSTRPROCID OR
                    TYPEV <BOOSTRPROCID
                    THEN FLAG(004);
                    IF NOT SPECTOG THEN
                        FUNCTOG#TRUE;
                        CHKS08
                    END
                END
            ELSE
                IF G#SAVEV OR G#0 THEN TYPEV#PROCID
                ELSE
                    IF TYPEV#REALSTRPROCID+G<BOOPROCID OR TYPEV>INTPROCID
                    THEN FLAG(005)
                    ELSE BEGIN FUNCTOG#TRUE;G#GTA1[J+J-1];
                        END;
                IF NOT STREAMTOG THEN SEGMENTSTART(G#SAVEV);
                SAV # G#SAVEV;

```

```

14259110 T 0029
14259120 T 0031
14259130 T 0034
14259140 T 0035
14259150 T 0035
14259160 T 0038
14260000 T 0038
14261000 T 0038
14262000 T 0038
14263000 T 0039
14264000 T 0040
14265000 T 0041
14265900 T 0041
14266000 T 0042
14266100 T 0043
14267000 T 0043
14268000 T 0043
14269000 T 0045

```

```

98 IS 48 LONG, NEXT SEG 94
T 0120
14270000 T 0120
14271000 T 0121
14272000 T 0121

```

```

START OF SEGMENT ***** 99
14273000 T 0000
14274000 T 0000

14275000 T 0000
14276000 T 0001
14277000 T 0002
14278000 T 0003
14279000 T 0005
14280000 T 0005
14281000 T 0007
14282000 T 0010
14283000 T 0010
14284000 T 0011
14285000 T 0013
14286000 T 0013
14287000 T 0015
14288000 T 0015
14289000 T 0017
14290000 T 0017
14291000 T 0017
14292000 T 0017
14293000 T 0020
14294000 T 0021
14295000 T 0023
14296000 T 0025
14297000 T 0028
14298000 T 0028
14299000 T 0031
14300000 T 0032
14301000 T 0032
14302000 T 0032

```



```

MODE←MODE+1;
LO←PROINFO;
SCATTERELBAT;
COMMENT CHECK TO SEE IF DECLARED FORWARD PREVIOUSLY
IF LEVEL=LEVEL
THEN
BEGIN
IF G+TAKE(LINKF+1)≥0
THEN FLAG(006);
FWOTOG←TRUE;
PROAD←ADDRSF;
PROINFO←ELBAT[I]; MARK←LINKF+INCRF; STEPIT
; PUT(=G, LINKF+1);
END
ELSE
BEGIN STOPENTRY←TRUE; P2←TRUE;
STOPGSP←LEVEL>1 AND STREAMTOG;
ENTRY(TYPEV); MARK←NEXTINFO; PUTNBUMP(0);
STOPGSP←FALSE;
PROINFO←TAKE(LASTINFO)& LASTINFO[35:35:13]; PROAD←ADDRSF;
P2←STOPENTRY←FALSE
END;
PJ←0; LEVEL←LEVEL+1;
IF STREAMTOG THEN STREAMWORDS;
IF ELCLASS=SEMICOLON THEN GO TO START1;
IF ELCLASS≠LEFTPAREN THEN FLAG(007);
COMMENT: THE FOLLOWING 8 STATEMENTS FOOL THE SCANNER AND BLOCK, PUTTING
FORMAL PARAMETER ENTRIES IN THE ZERO ROW OF INFO;
RR1←NEXTINFO;
LASTINFOT←LASTINFO; LASTINFO←NEXTINFO+1;
PUTNBUMP(0);
PTOG←TRUE; I←I+1;
ENTRY(SECRET);
IF FWOTOG THEN
BEGIN
IF GT1:=TAKE(MARK), [40:8] ≠ PJ THEN FLAG(48); % WRONG
% NUMBER OF PARAMETERS. WE DON'T WANT TO Clobber INFO.
END
ELSE
PUT(PJ, MARK);
P←PJ;
IF ELCLASS≠RTPAREN
THEN FLAG(008);
IF STEPI≠SEMICOLON
THEN FLAG(009);
COMMENT MARK PARAMETERS VALUE IF THERE IS A VALUE PART;
IF STEPI=VALUEV
THEN
BEGIN
DO
IF STEPI≠SECRET
THEN FLAG(010)
ELSE
BEGIN
IF G+ELBAT[I].ADDRESS=0 OR G>PJ
THEN
FLAG(010);

```

```

14303000 T 0032
14304000 T 0033
14305000 T 0034
14306000 T 0034
14307000 T 0034
14308000 T 0035
14309000 T 0035
14310000 T 0036
14311000 T 0037
14312000 T 0039
14313000 T 0040
14314000 T 0041
14315000 T 0043
14316000 T 0045
14317000 T 0045
14318000 T 0045
14318500 T 0047
14319000 T 0049
14319500 T 0051
14320000 T 0052
14321000 T 0055
14322000 T 0055
14323000 T 0056
14324000 T 0058
14325000 T 0059
14326000 T 0061
14327000 T 0063
14328000 T 0063
14329000 T 0063
14330000 T 0063
14331000 T 0065
14332000 T 0066
14333000 T 0068
14333100 T 0069
14333200 T 0069
14333300 T 0070
14333400 T 0073
14333500 T 0073
14333600 T 0073
14334000 T 0073
14335000 T 0075
14336000 T 0075
14337000 T 0076
14338000 T 0077
14339000 T 0078
14340000 T 0080
14341000 T 0080
14342000 T 0080
14343000 T 0081
14344000 T 0081
14345000 T 0082
14346000 T 0082
14347000 T 0083
14348000 T 0084
14349000 T 0084
14350000 T 0087
14351000 T 0087

```

```

                                G+TAKE(ELBAT[I]);
                                PUT(G&1[10:47:1],ELBAT[I]);
                                END
                                UNTIL
                                  STEPI#COMMA;
IF ELCLASS#SEMICOLON
  THEN FLAG(011)
  ELSE STEPI
    END;I+I-1;
IF STREAMTOG
  THEN
    BEGIN
      BUP#PJ; SPECTOG#TRUE;GO TO START1
    END
  ELSE
    BEGIN
      SPECTOG#TRUE;
      BUP#0;
      IF ELCLASS#DECLARATORS
        THEN FLAG(012)
      END;
      START1#PTOG#FALSE;LASTINFO#LASTINFOT;NEXTINFO#IF FWDTOG THEN RR1 ELSE
        MARK#PJ+1;
      START1#PINFO#NEXTINFO;
      START2#END;
PRT(704) = *SEGMENT DESCRIPTOR*

      IF SPECTOG OR STREAMTOG
        THEN
          GO TO START;
      COMMENT IF SPECTOG IS ON THEN THE BLOCK WILL PROCESS THE SPECIFICATION
        PART SIMILARY TO DECLARATIONS WITH A FEW NECESSARY VARIATIONS;
      HF;
      BEGIN
        LABEL START,STOP;
PRT(705) = *SEGMENT DESCRIPTOR*

        DEFINE TESTLEV = LEVEL>2 #;
IF STREAMTOG
  THEN BEGIN
    IF TESTLEV THEN JUMPCHKNX ELSE SEGMENT$START(TRUE);PJ#P;
      PTOG#FALSE;
      PUT(TAKE(GIT(PROINFO))&L[28:36:12],GIT(PROINFO));
    IF TESTLEV THEN BEGIN EMITO(584); END;
      IF STEPI#BEGINV
        THEN
          BEGIN
            WHILE STEPI#DECLARATORS OR ELCLASS#LOCALV
              DO
                BEGIN
                  IF ELBAT[I].ADDRESS#LABELV
                    THEN
                      BEGIN
                        STOPDEFINE#STOPGSP#STOPENTRY#TRUE;
DO BEGIN STOPDEFINE#TRUE;STEPI;ENTRY(STLABID);PUTNBUMP(0) END UNTIL
          ELCLASS#COMMA;STOPGSP#STOPENTRY#FALSE
            END

```

```

14352000 T 0089
14353000 T 0090
14354000 T 0092
14355000 T 0092
14356000 T 0092
14357000 T 0094
14358000 T 0094
14359000 T 0095
14360000 T 0096
14361000 T 0098
14362000 T 0098
14363000 T 0098
14364000 T 0099
14365000 T 0101
14366000 T 0101
14367000 T 0101
14368000 T 0101
14369000 T 0102
14370000 T 0103
14371000 T 0103
14372000 T 0104
14373000 T 0105
14374000 T 0109
14375000 T 0110
14376000 T 0111

```

99 IS 113 LONG, NEXT SEG 94

```

14377000 T 0122
14378000 T 0122
14379000 T 0122
14380000 T 0123
14381000 T 0123
14382000 T 0123
14383000 T 0124
14384000 T 0124

```

START OF SEGMENT ***** 100

```

14384100 T 0000
14385000 T 0000
14386000 T 0000
14387000 T 0000
14388000 T 0004
14388100 T 0005
14389000 T 0008
14393000 T 0010
14394000 T 0011
14395000 T 0011
14396000 T 0012
14397000 T 0014
14398000 T 0015
14399000 T 0015
14400000 T 0016
14401000 T 0017
14402000 T 0017
14403000 T 0019
14404000 T 0022
14405000 T 0024

```

```

ELSE
  BEGIN
    I=I+1;
    ENTRY(LOCLID)
  END
END;
IF FUNCTOG THEN
  PUT((Z+TAKE(PROINFO))&LOCLID[2:41:7] &
    (PJ+2+REAL(TESTLEV))[16:37:11],PROINFO);
  COMPOUNDTAIL
END
ELSE
  BEGIN
    IF FUNCTOG THEN
      PUT((Z+TAKE(PROINFO))&LOCLID[2:41:7]&
        (PJ+2+REAL(TESTLEV))[16:37:11],PROINFO);
    STREAMSTMT;
  END;
  COMMENT THE FOLLOWING BLOCK CONSTITUTES THE STREAM PROCEDURE PURGE;
  BEGIN
    REAL NLOC,NLAB;
    PRT(706) = *SEGMENT DESCRIPTOR*
    STACK(F+25) = NLOC
    STACK(F+26) = NLAB

```

```

  DEFINE SES=18#,SED=6#,TRW=5#;
  DEFINE LOC=[36:12]#,LASTGT=[24:12]#;
  J← LASTINFO;
  NLOC←NLAB+0;
  DO
  BEGIN
    IF(GT1+TAKE(J)).CLASS=LUCLID THEN
      BEGIN
        IF BOOLEAN(GT1.FORMAL) THEN
          BEGIN
            IF GT1<0 THEN
              PUT(TAKE(GT2+MARK+P=GT1,ADDRESS+1)&FILEID[2:41:7]
                ,GT2);
            END
          ELSE NLOC←NLOC+1;
          END
        ELSE
          BEGIN
            IF GT1.ADDRESS≠0 THEN NLAB←NLAB+1;
            IF(GT3+TAKE(GIT(J))).LASTGT≠0 AND GT3.LOC =0 THEN
              BEGIN
                MOVE(9,INFO[0,J],ACCUM[0]);
                Q←ACCUM[1];
                FLAG(267);
                ERRORTOG←TRUE;
              END;
            END;
          G←(GT2+TAKE(J+1)).PURPT;
          IF GT1.[2:8] ≠ STLABID×2+1 THEN
            STACKHEAD[(0&GT2[12:12:36])MOD 125]+TAKE(J).LINK;
          END UNTIL J=J+G$1;

```

```

14406000 T 0025
14407000 T 0025
14408000 T 0025
14409000 T 0027
14410000 T 0027
14411000 T 0027
14411100 T 0028
14411200 T 0028
14411300 T 0031
14412000 T 0034
14413000 T 0034
14414000 T 0035
14415000 T 0035
14415100 T 0035
14415200 T 0035
14415300 T 0038
14415400 T 0041
14415500 T 0042
14416000 T 0042
14417000 T 0042
14418000 T 0042

```

START OF SEGMENT ***** 101

```

14419000 T 0000
14420000 T 0000
14421000 T 0000
14422000 T 0000
14423000 T 0002
14424000 T 0002
14425000 T 0002
14426000 T 0004
14427000 T 0004
14428000 T 0005
14429000 T 0006
14430000 T 0006
14431000 T 0011
14432000 T 0012
14433000 T 0012
14434000 T 0014
14435000 T 0014
14436000 T 0014
14437000 T 0014
14438000 T 0017
14439000 T 0021
14440000 T 0022
14441000 T 0024
14442000 T 0025
14443000 T 0026
14444000 T 0027
14445000 T 0027
14446000 T 0027
14447000 T 0030
14448000 T 0032
14449000 T 0036
14450000 T 0038

```

```

        IF TESTLEV THEN BEGIN EMITC(1,0); EMITC(BFW) END
        ELSE EMIT(0);
PUT(TAKE(MARK)&NLOC[1:42:6]&L[16:36:12]&P[40:40:8],MARK);
        IF FUNCTOG THEN
            PUT(Z, PROINFO);
        STREAMWORDS;
        STREAMTOG+FALSE;
        IF NOT TESTLEV THEN BEGIN PROGDESCBLDR(PROAD,TRUE,(L+3)DIV 4,CHAR);
            SEGMENT((L+3)DIV 4,PROINFO,ADDRESS);
            RIGHT(L); L+0;
        END;
        IF LISTER AND FORMATOG THEN SPACEITDOWN;
        END;
PRT(707) = *SEGMENT DESCRIPTOR*

```

```

14451000 T 0038
14451100 T 0041
14451200 T 0043
14452000 T 0047
14457000 T 0047
14460000 T 0049
14461000 T 0049
14461100 T 0050
14461200 T 0054
14461300 T 0056
14461400 T 0058
14461500 T 0058
14462000 T 0068

```

```

        LASTINFO+LASTINFOT;NEXTINFO+MARK+P+1;
        END
    ELSE
        BEGIN
            IF STEPI=FORWARDV
            THEN
                BEGIN
                    PUT(=TAKE(G+PROINFO,LINK+1),G);
                    PURGE(PINFO);
                STEPIT
                END
            ELSE
                BEGIN
                    PROADO+PROAD;
                    TSUBLEVEL+TSUBLEVEL;SUBLEVEL+LEVEL;STACKCTRO+STACKCTR;
                    IF MODE=1 THEN FRSTLEVEL+LEVEL;STACKCTR+513+REAL(FUNCTOG);
                    IF ELCLASS = BEGINV THEN
                        BEGIN
                            CALLINFO+(CALLX+CALLX+1)+1;
                            NESTCTR+STACKCTR;
                            BLOCK(TRUE)
                            ; PURGE(PINFO);
                            IF NESTOG THEN
                                BEGIN GT1+TAKE(PROINFO,ADDRESS);
                                    NESTPRT[GT1]+0&PROINFO[35:35:13]&CALLINFO[22:35:13];
                                    CALL[CALLINFO-1]+(TAKE(GIT(PROINFO))+NESTCTR-511)&
                                        CALLX[22:35:13];
                                END;
                                L+0;
                                GO TO STOP END;
                                BEGIN
                                    FLAG(052);
                                    RELAD+L ;
                                    STMT;
                                    HTTEOAP(FALSE,RELAD,PINFO,PROAD);
                                    END;
                                STOP;
                                SUBLEVEL+TSUBLEVEL;
                                STACKCTR+STACKCTRO;
                                IF LISTER AND FORMATOG THEN SPACEITDOWN;
                                END;
                                END;

```

```

101 IS 69 LONG, NEXT SEG 100
14463000 T 0043
14464000 T 0045
14465000 T 0045
14466000 T 0045
14467000 T 0046
14468000 T 0046
14469000 T 0047
14470000 T 0047
14471000 T 0050
14472000 T 0051
14473000 T 0051
14474000 T 0052
14475000 T 0052
14476000 T 0052
14477000 T 0053
14478000 T 0055
14479000 T 0058
14481000 T 0059
14481100 T 0060
14481200 T 0062
14482000 T 0063
14483000 T 0063
14483100 T 0064
14483200 T 0064
14483300 T 0067
14483400 T 0070
14483500 T 0076
14483600 T 0077
14483700 T 0077
14484000 T 0078
14485000 T 0079
14486000 T 0079
14487000 T 0079
14488000 T 0080
14489000 T 0081
14490000 T 0082
14491000 T 0082
14492000 T 0083
14493000 T 0083
14493500 T 0084
14494000 T 0094
14495000 T 0094

```

```

        PROINFO+LO;
        IF JUMPCTR=LEVEL
        THEN
            JUMPCTR+LEVEL-1;
            LEVEL+LEVEL-1;
            MODE+MODE-1;
            MAXSTACK+MAXSTACK;
        START;END;
PRT(710) = *SEGMENT DESCRIPTOR*

        GO TO START;
        CALLSTATEMENT; FOULED + L;
        JUMPCHKX; IF SOP THEN BEGIN Z+STACKCTR-513; WHILE Z+Z=120
            DO EMITL(0) END;
        IF SPECTOG THEN BEGIN
            FLAG(12); GO TO HF
        END;
        BEGINCTR + BEGINCTR-1;
        IF ERRORTOG
        THEN COMPOUNDTAIL
        ELSE
            BEGIN
                STMT;
                IF ELCLASS+TABLE(I+1)=DECLARATORS
                THEN
                    BEGIN
                        ELBAT[I].CLASS+SEMICOLON;
                        BEGINCTR+BEGINCTR+1;
                        GO TO START
                    END
                ELSE
                    COMPOUNDTAIL
            END;
        FUNCTOG+FUNCTOG;
        IF SOP THEN HTTEDAP(FALSE,FIRSTX,NINFOO,BLKAD)
        ELSE BEGIN IF NESTOG THEN SORTNEST; PURGE(NINFOO); END;
        SEGMENT((L+3)DIV 4,PROADD);
        IF LEVEL>1 THEN RIGHT(L);
        IF LEVEL + LEVEL-1 = 0 THEN CONSTANTCLEAN;

        AJUMP+AJUMP;

        FIRSTX+FIRSTX;
        SAVEL+SAVEL;
        STACKCTR+STACKCTR;

END BLOCK;

```

COMMENT THIS SECTION CONTAINS THE VARIABLE ROUTINE AND ITS SIDEKICKS;

14496000	T	0094	
14497000	T	0095	
14498000	T	0095	
14499000	T	0095	
14500000	T	0097	
14501000	T	0098	
14502000	T	0100	
14503000	T	0100	
100 IS	102 LONG,	NEXT SEG	94
14504000	T	0125	
14505000	T	0125	
14506000	T	0126	
14506500	T	0131	
14507000	T	0133	
14508000	T	0134	
14509000	T	0135	
14510000	T	0135	
14511000	T	0136	
14512000	T	0136	
14513000	T	0137	
14514000	T	0138	
14515000	T	0138	
14516000	T	0139	
14517000	T	0140	
14518000	T	0141	
14519000	T	0141	
14520000	T	0144	
14521000	T	0145	
14522000	T	0146	
14523000	T	0146	
14524000	T	0146	
14525000	T	0146	
14599000	T	0147	
14600000	T	0147	
14601000	T	0149	
14602000	T	0152	
14603000	T	0154	
14604000	T	0156	
14605000	T	0159	
14606000	T	0159	
14607000	T	0160	
14608000	T	0160	
14609000	T	0160	
14610000	T	0161	
14611000	T	0162	
14612000	T	0162	
14613000	T	0162	
94 IS	170 LONG,	NEXT SEG	3

15000000	T	0586
15001000	T	0586
15002000	T	0586
15003000	T	0586

COMMENT THE FOLLOWING BLOCK HANDLES THE FOLLOWING CASES
OF SIMPLE VARIABLES:

1. V + EXP ,WHERE V IS FORMAL-CALL BY NAME.
2. V * EXP ,ALL V EXCEPT FORMAL-NAME.
3. V.[S:L] + EXP ,WHERE V IS FORMAL-CALL BY NAME.
4. V.[S:L] * EXP ,ALL V EXCEPT FORMAL-NAME.
5. V.[S:L] ,ALL V.
6. V ,ALL V.

CODE EMITTED FOR THE ABOVE CASES IS AS FOLLOWS:

1. VN,EXP,M*,XCH,+,
2. EXP,M*,VL,+,
3. VN,DUP,COC,EXP,T,M*,XCH,+,
4. VV,EXP,T,M*,VL,+,
5. ZEROL,VV,T ,
6. VV .

WHERE VN = DESC V
EXP = ARITH. OR BOOLEAN EXPRESSION,AS REQUIRED.
M* = CALL ON MONITOR ROUTINE,IF REQUIRED.
VL = LITC V
VV = OPDC V
+ = STORE INSTRUCTION(ISO,ISN,SD OR STD).
T = BIT TRANSFER CODE(DIA,DIB,TRB).
ZEROL = LITC 0
DUP,COC,XCH = THE INSTRUCTIONS DUP,COC,AND XCH.

15004000	T	0586
15005000	T	0586
15006000	T	0586
15007000	T	0586
15008000	T	0586
15009000	T	0586
15012000	T	0586
15013000	T	0586
15014000	T	0586
15015000	T	0586
15016000	T	0586
15017000	T	0586
15018000	T	0586
15019000	T	0586
15020000	T	0586
15021000	T	0586
15022000	T	0586
15023000	T	0586
15024000	T	0586
15025000	T	0586
15026000	T	0586
15027000	T	0586
15028000	T	0586
15029000	T	0586
15030000	T	0586
15031000	T	0586
15032000	T	0586
15033000	T	0586
15034000	T	0586
15035000	T	0586
15036000	T	0586
15037000	T	0586
15038000	T	0586
15039000	T	0586
15040000	T	0586
15041000	T	0586
15042000	T	0586
15043000	T	0586
15044000	T	0586
15045000	T	0586
15046000	T	0586
15047000	T	0586
15048000	T	0586
15049000	T	0586
15050000	T	0586
15051000	T	0586
15052000	T	0586
15053000	T	0586
15054000	T	0586
15055000	T	0586
15056000	T	0586
15057000	T	0586
15058000	T	0586
15059000	T	0586
15060000	T	0586
15061000	T	0586
15062000	T	0586

OF COURSE, EXP WILL CAUSE RECURSION, IN GENERAL, AND THUS
THE PARAMETER P1 AND THE LOCALS CAN NOT BE HANDLED IN A
GLOBAL FASHION.
THE PARAMETER P1 IS USED TO TELL THE VARIABLE ROUTINE
WHO CALLED IT. SOME OF THE CODE GENERATION AND SOME
SYNTAX CHECKS DEPEND UPON A PARTICULAR VALUE OF P1 ,
;

PROCEDURE VARIABLE(P1); INTEGER P1;

BEGIN
REAL TALL, COMMENT ELBAT WORD FOR VARIABLE;

STACK(F+2) = TALL

STACK(F+3) = T1

STACK(F+4) = T2

STACK(F+5) = J

STACK(F+6) = SPCLMON

T1 , COMMENT 1ST INTEGER OF PARTIAL WORD SYNTAX;

T2 , COMMENT 2ND INTEGER OF PARTIAL WORD SYNTAX;

J ; COMMENT SUBSCRIPT COUNTER ;

LABEL EXIT,L1, LAST, NEXT, JAZZ, ITUP, LASS;
DEFINE FORMALNAME=[9:2]=2#, LONGID=NAMEID#;
BOOLEAN SPCLMON;

TALL=ELBAT[I] ;
IF ELCLASS ≤ INTPROCID THEN
BEGIN

COMMENT 211 VARIABLE=FUNCTION IDENTIFIER USED OUTSIDE OF ITS SCOPE*;
IF TALL.LINK ≠ PROINFO.LINK THEN
BEGIN ERR(211); GO TO EXIT END;
TALL=TALL & (ELCLASS+4)[2:41:7] & 513[16:37:11];
END
ELSE CHECKER(TALL);
IF TALL.CLASS ≤ INTID THEN
BEGIN

IF STEP1= ASSIGNOP THEN
BEGIN STACKCT + 1;
L1: IF TALL.FORMALNAME THEN
BEGIN
EMITN(TALL.ADDRESS);
IF T1≠0 THEN BEGIN EMITD(DUP); EMITD(COC) END;
END
ELSE IF T1≠0 THEN EMITV(TALL.ADDRESS)
; STACKCT + REAL(T1≠0); STEPIT;
AEXP;
EMITD(48-T2, T1, T2);

STACKCT + 0;
GT1 + IF TALL.CLASS = INTID THEN IF P1= FS
THEN ISD ELSE ISN ELSE
IF P1 = FS THEN STD ELSE SND ;

IF TALL.FORMALNAME THEN
BEGIN
EMITD(XCH); IF TALL.ADDRESS>1023 THEN EMITD(PRTE);
EMITD(GT1);
END
ELSE EMITPAIR(TALL.ADDRESS, GT1);

15063000 T 0586
15064000 T 0586
15065000 T 0586
15066000 T 0586
15067000 T 0586
15068000 T 0586
15069000 T 0586
15070000 T 0586
15071000 T 0586
15072000 T 0586

START OF SEGMENT ***** 102

15073000 T 0000
15074000 T 0000
15075000 T 0000
15076000 T 0000
15076100 T 0000
15076200 T 0000
15077000 T 0000
15078000 T 0001
15079000 T 0001
15080000 T 0002
15081000 T 0004
15082000 T 0005
15083000 T 0005
15084000 T 0009
15085000 T 0009
15086000 T 0010
15087000 T 0011
15088000 T 0012
15089000 T 0012
15090000 T 0012
15091000 T 0013
15092000 T 0014
15093000 T 0016
15094000 T 0016
15095000 T 0018
15096000 T 0020
15097000 T 0020
15098000 T 0022
15099000 T 0025
15100000 T 0026
15101000 T 0027
15101500 T 0027
15102000 T 0028
15103000 T 0030
15104000 T 0033
15105000 T 0035
15106000 T 0037
15106100 T 0037
15106200 T 0040
15106300 T 0041
15107000 T 0041

```

END
ELSE
  BEGIN
    IF P1=FL THEN BEGIN
      IF ELCLASS < AMPERSAND THEN EMITN(TALL,ADDRESS)
      ELSE EMITV(TALL,ADDRESS);
      GO TO EXIT END;
    IF ELCLASS= PERIOD THEN
      BEGIN IF DOTSYNTAX(T1,T2) THEN GO TO EXIT ;
      IF STEPI=ASSIGNOP THEN
        IF P1≠ FS THEN
          BEGIN ERR(201);GO TO EXIT END
          ELSE GO TO L1
        END ;
      IF P1≠ FP THEN BEGIN ERR(202); GO TO EXIT END;
    COMMENT 202 VARIABLE= A VARIABLE APPEARS WHICH IS NOT FOLLOWED *
      BY A LEFT ARROW OR PERIOD *;
    COMMENT 201 VARIABLE= A PARTIAL WORD DESIGNATOR IS NOT THE *
      LEFT-MOST OF A LEFT PART LIST *;
      EMITI(TALL,T1,T2);
    END ;
  END OF SIMPLE VARIABLES
ELSE
  IF TALL.CLASS≠LABELID THEN
    COMMENT THE FOLLOWING BLOCK HANDLES THESE CASES OF SUBSCRIPTED
      VARIABLES:
      1. V[*] ,ROW DESIGNATOR FOR SINGLE-DIMENSION.
      2. V[R,*] ,ROW DESIGNATOR FOR MULTI-DIMENSION.
      3. V[R] ,ARRAY ELEMENT,NAME OR VALUE.
      4. V[R],[S:L] ,PARTIAL WORD DESIGNATOR, VALUE.
      5. V[R] * ,ASSIGNMENT TO ARRAY ELEMENT.
      6. V[R],[S:L] * ,ASSIGNMENT TO PARTIAL WORD,LEFT-MOST.
    R IS A K-ORDER SUBSCRIPT LIST,I.E. R= R1,R2,...,RK.
    IN THE CASE OF NO MONITORING ON V, THE FOLLOWING CODE
    IS EMITTED FOR THE ABOVE CASES:
      1. CASE #1 IS A SPECIAL CASE OF #2,NAMELY,SINGLE
        DIMENSION. THE CODE EMITTED IS:
        VL,LOD.
        EXECUTION: PLACES ARRAY DESCRIPTER IN REG A.
      2. THIS CODE IS BASIC TO THE SUBSCRIPTION PROCESS.
        EACH SUBSCRIPT GENERATES THE FOLLOWING SEQUENCE
        OF CODE:
        AEXP,L*,IF FIRST SUBSCRIPT THEN VN ELSE CDC
        ,LOD.
        FOR A K-ORDER SUBSCRIPTION,K-1 SEQUENCE ARE
        PRODUCED, THE AEXP IN EACH SEQUENCE REFERS TO
        THE CODE PRODUCED BY THE ARITHMETIC EXPRESSION
        PROCEDURE FOR THE ACTUAL SUBSCRIPT EXPRESSIONS,
        [* REFERS TO THE CODE PRODUCED FOR SUBTRACTING
        NON-ZERO LOWER BOUNDS FROM THE SUBSCRIPT
        EXPRESSION(L* YIELDS NO CODE FOR ZERO BOUNDS).
        EXECUTION: PLACES ARRAY ROW DESCRIPTOR IN REG A
        , THE SPECIFIC ROW DEPENDS UPON THE
        VALUES OF THE K-1 SUBSCRIPTS.
    FOR THE REMAINING CASES,

```

15108000	T	0043
15109000	T	0043
15110000	T	0043
15110100	T	0044
15110200	T	0045
15110300	T	0046
15110400	T	0049
15111000	T	0050
15112000	T	0050
15113000	T	0053
15114000	T	0054
15115000	T	0055
15116000	T	0057
15117000	T	0057
15118000	T	0057
15119000	T	0057
15120000	T	0059
15121000	T	0059
15122000	T	0059
15123000	T	0059
15124000	T	0059
15125000	T	0060
15126000	T	0060
15127000	T	0060
15128000	T	0060
15128100	T	0060
15129000	T	0062
15130000	T	0062
15131000	T	0062
15132000	T	0062
15133000	T	0062
15134000	T	0062
15135000	T	0062
15136000	T	0062
15137000	T	0062
15138000	T	0062
15139000	T	0062
15140000	T	0062
15141000	T	0062
15142000	T	0062
15143000	T	0062
15144000	T	0062
15145000	T	0062
15146000	T	0062
15147000	T	0062
15148000	T	0062
15149000	T	0062
15150000	T	0062
15151000	T	0062
15152000	T	0062
15153000	T	0062
15154000	T	0062
15155000	T	0062
15156000	T	0062
15157000	T	0062
15158000	T	0062
15159000	T	0062


```

SEQUENCES OF CODE ARE EMITTED AS IN CASE #2.
HOWEVER, THE ACTUAL SEQUENCES ARE:
ONE SEQUENCE, (AEXP, L*), FOR THE 1ST SUBSCRIPT,
K=1 SEQUENCES, (IF FIRST SUBSCRIPT THEN VN
ELSE CDC, LOD, AEXP, L*), FOR THE REMAINING
SUBSCRIPTS, IF K>1.
AT THIS POINT, CASES #3-6 ARE DIFFERENTIATED
AND ADDITION CODE, PARTICULAR TO EACH CASE, IS
EMITTED.
3. ADD THE SEQUENCE:
   IF FIRST SUBSCRIPT THEN VV ELSE CDC.
EXECUTION: THE ARRAY ELEMENT IS PUT IN REG A.
4. ADD THE SEQUENCE:
   IF FIRST SUBSCRIPT THEN VV ELSE CDC, ZEROL,
   XCH, T.
5. ADD THE SEQUENCE:
   IF FIRST SUBSCRIPT THEN VN ELSE CDC, EXP,
   XCH, +.
6. ADD THE SEQUENCE:
   IF FIRST SUBSCRIPT THEN VN ELSE CDC, DUP, LOD,
   EXP, T, XCH, +.
EXP, T, +, ZEROL, ETC. HAVE SAME MEANINGS AS DEFINED IN
SIMPLE VARIABLE BLOCK.
BEGIN

IF STEPI ≠ LFTBRKET THEN
BEGIN
IF ELCLASS = PERIOD THEN
BEGIN
IF DOTSYNTAX(T1, T2) THEN GO TO EXIT;
IF STEPI = ASSIGNOP THEN
BEGIN
IF P1≠FS THEN BEGIN ERR(209); GO EXIT END;
IF TALL, CLASS ≤ INTARRAYID THEN
BEGIN EMITPAIR(TALL, ADDRESS, LOD) END
ELSE EMITN(TALL, ADDRESS); STACKCT + STACKCT+1;
JAZZ: STEPIT; AEXP;
EMITD(48-T2, T1, T2);
EMITPAIR(TALL, ADDRESS,
        IF P1=FS THEN STD ELSE SND);
STACKCT + 0; END
ELSE BEGIN
ITUP: EMITI(TALL, T1, T2);

        END;
GO TO EXIT ;
END;
IF ELCLASS = ASSIGNOP THEN GO TO JAZZ ELSE GO TO ITUP ;
END;
J + 0;

```

```

15160000 T 0062
15161000 T 0062
15162000 T 0062
15163000 T 0062
15164000 T 0062
15165000 T 0062
15166000 T 0062
15167000 T 0062
15168000 T 0062
15169000 T 0062
15170000 T 0062
15171000 T 0062
15172000 T 0062
15173000 T 0062
15174000 T 0062
15175000 T 0062
15176000 T 0062
15177000 T 0062
15178000 T 0062
15179000 T 0062
15180000 T 0062
15181000 T 0062
15182000 T 0062
15183000 T 0062
15184000 T 0063
15184100 T 0063
15184200 T 0063
15184300 T 0063
15184400 T 0063
15233000 T 0063
15233002 T 0064
15233003 T 0064
15233004 T 0065
15233005 T 0065
15233006 T 0067
15233007 T 0068
15233008 T 0069
15233009 T 0071
15233010 T 0072
15233011 T 0074
15233012 T 0077
15233013 T 0079
15233014 T 0080
15233015 T 0081
15233016 T 0084
15233017 T 0085
15233018 T 0085
15233019 T 0087
15233020 T 0087
15233021 T 0087
15233022 T 0087
15233023 T 0087
15233024 T 0087
15233025 T 0087
15233026 T 0087
15233027 T 0089
15234000 T 0089

```

	STACKCT + 0;	15234500	T	0090
COMMENT 207	VARIABLE=MISSING LEFTBRACKET ON SUBSCRIPTED VARIABLE *;	15235000	T	0091
NEXT;	IF STEPI = FACTOP THEN	15253000	T	0091
	BEGIN	15254000	T	0092
	IF J+1 > TALL, INCR THEN	15255000	T	0092
	BEGIN ERR(203); GO EXIT END;	15256000	T	0094
COMMENT 203	VARIABLE= THE NUMBER OF SUBSCRIPTS USED IN A ROW *	15257000	T	0096
	ROW DESIGNATER DOES NOT MATCH THE ARRAY *	15258000	T	0096
	DECLARATION, *	15259000	T	0096
	IF STEPI < RTBRKET THEN	15260000	T	0096
	BEGIN ERR(204); GO EXIT END;	15261000	T	0097
COMMENT 204	VARIABLE= COMPILER EXPECTS A J IN A ROW DESIGNATER *;	15262000	T	0098
COMMENT 205	VARIABLE= A ROW DESIGNATER APPEARS OUTSIDE OF A FILL *	15263000	T	0098
	STATEMENT OR ACTUAL PARAMETER LIST, *	15264000	T	0098
	IF J=0 THEN	15265000	T	0098
	EMITPAIR(TALL, ADDRESS, LOD);	15266000	T	0098
	STLB=0;	15267000	T	0099
	STEPIT;	15273000	T	0101
	GO TO EXIT;	15274000	T	0102
	END OF ROW DESIGNATOR PORTION ;	15275000	T	0102
	IF ELCLASS=LITND AND ELBAT(I), ADDRESS=0 AND TABLE(I+1)=RTBRKET	15276000	T	0103
	AND TALL, CLASSNAMEID THEN	15276010	T	0103
	BEGIN	15276020	T	0107
	I=I+1;	15276030	T	0109
	IF STEPI=ASSIGNOP THEN BEGIN	15276040	T	0109
LASS:	IF T1=0 THEN EMITV(TALL, ADDRESS);	15276050	T	0111
	STEPIT; AEXP; EMITD(48=T2, T1, T2);	15276060	T	0112
	EMITN(TALL, ADDRESS);	15276070	T	0115
	EMITD(IF TALL, CLASSNAMEID THEN	15276080	T	0118
	IF P1=FS THEN ISD ELSE ISN ELSE	15276090	T	0119
	IF P1=FS THEN STD ELSE SND);	15276100	T	0121
	STACKCT + 0;	15276110	T	0124
	GO TO EXIT END	15276115	T	0126
	ELSE	15276120	T	0127
	IF ELCLASS = PERIOD THEN BEGIN	15276130	T	0128
	IF DOTSYNTAX(T1, T2) THEN GO TO EXIT;	15276140	T	0128
	IF STEPI = ASSIGNOP THEN IF P1=FS THEN GO TO LASS	15276150	T	0129
	ELSE BEGIN ERR(209); GO EXIT END;	15276160	T	0131
	END;	15276170	T	0133
	IF P1=FS THEN BEGIN ERR(210); GO EXIT END;	15276180	T	0135
	EMITI(IF P1=FL THEN TALL ELSE TALL&REALID(2141:7), T1, T2);	15276190	T	0135
	GO TO EXIT;	15276200	T	0138
	END;	15276210	T	0138
	AEXP;	15276220	T	0142
	STACKCT + 1;	15276230	T	0142
	J + J + 1;	15276240	T	0142
	IF ELCLASS = COMMA THEN	15277000	T	0142
	BEGIN	15278000	T	0143
COMMENT *****	MONITOR FUNCTION M4 GOES HERE ;	15280000	T	0144
	IF J = 1 THEN EMITV(TALL, ADDRESS) ELSE EMITD(COC);	15287000	T	0145
	GO TO NEXT;	15288000	T	0146
	END OF SUBSCRIPT COMMA HANDLER ;	15289000	T	0146
	IF ELCLASS < RTBRKET THEN BEGIN ERR(206); GO EXIT END;	15290000	T	0146
		15291000	T	0150
		15292000	T	0150
		15293000	T	0150
		15294000	T	0150

COMMENT 206	VARIABLE= MISSING RIGHT BRACKET ON SUBSCRIPTED VARIABLE*; GT1+IF TALL.CLASSNAMEID THEN 1 ELSE TALL.INCR; IF J#GT1 THEN	15295000 T 0153
	BEGIN ERR(208);GO TO EXIT END;	15295100 T 0153
COMMENT 208	VARIABLE= NUMBER OF SUBSCRIPTS DOES NOT MATCH WITH * ARRAY DECLARATION. *; IF STEPI = ASSIGNOP THEN	15296000 T 0157
	BEGIN	15297000 T 0157
LAST:	IF J=1 THEN EMITN(TALL.ADDRESS) ELSE EMITO(CDC);	15298000 T 0159
	IF TALL.CLASS ≥ LONGID THEN EMITO(INX);	15299000 T 0159
	IF T1= 0 THEN	15300000 T 0159
	BEGIN IF P1= FR THEN GO TO EXIT END	15301000 T 0160
	ELSE BEGIN EMITO(DUP);EMITO(LOD)END; STEPIT;	15302000 T 0161
	AEXP;	15303000 T 0164
	EMITD(48-T2,T1,T2) ;	15304000 T 0167
	EMITO(XCH);	15305000 T 0168
	IF TALL.ADDRESS>1023 THEN EMITO(PKTE);	15306000 T 0169
	EMITO(IF TALL.CLASS MOD 2 = INTARRAYID MOD 2 THEN	15307000 T 0172
	IF P1 = FS THEN ISD ELSE ISN ELSE	15308000 T 0172
	IF P1=FS THEN STD ELSE SND);	15309000 T 0174
	STACKCT + 0;	15310000 T 0175
	P1+0 ;	15333000 T 0177
	GO TO EXIT ;	15334000 T 0180
	END OF ASSIGNMENT STATEMENT SUBSCRIPTED VARIABLES;	15335000 T 0183
	IF ELCLASS=PERIOD THEN	15336000 T 0186
	BEGIN	15337000 T 0187
	IF DOTSYNTAX(T1,T2) THEN GO TO EXIT;	15338000 T 0188
	IF STEPI = ASSIGNOP THEN IF P1=FS THEN GO TO LAST	15339000 T 0188
	ELSE BEGIN ERR(209); GO EXIT END;	15340000 T 0189
	IF J#1 THEN EMITO(COC) ELSE IF TALL.CLASS ≥ LONGID THEN	15341000 T 0189
	BEGIN EMITN(TALL.ADDRESS);EMITO(INX);EMITO(LOD) END	15342000 T 0191
	ELSE EMITV(TALL.ADDRESS);	15343000 T 0193
	END	15344000 T 0195
	ELSE	15344100 T 0199
COMMENT *****	MONITOR FUNCTION M10 GOES HERE ;	15344200 T 0202
	BEGIN COMMENT MONITOR FUNCTION M10;	15345000 T 0204
	SPCLMON+P1 = FP OR ELCLASS ≥ AMPERSAND;	15346000 T 0204
	IF J = 1	15347000 T 0204
	THEN IF TALL.CLASS ≥ LONGID THEN	15348000 T 0204
	BEGIN	15349000 T 0204
	EMITN(TALL.ADDRESS); EMITO(INX);	15350000 T 0206
	IF SPCLMON THEN EMITO(LOD) ;	15351000 T 0207
	END ELSE IF SPCLMON	15351100 T 0209
	THEN EMITV(TALL.ADDRESS)	15351200 T 0209
	ELSE EMITN(TALL.ADDRESS)	15351300 T 0211
	ELSE EMITO(IF SPCLMON	15351400 T 0213
	THEN COC	15352000 T 0213
	ELSE CDC);	15353000 T 0214
	IF P1 =FS THEN ERR(210);	15354000 T 0216
	GO TO EXIT;	15355000 T 0218
	END;	15356000 T 0219
	IF P1=FS THEN BEGIN ERR(210); GO TO EXIT END ;	15364000 T 0220
COMMENT 210	VARIABLE=MISSING LEFT ARROW OR PERIOD. *; STACKCT + 0;	15365000 T 0222
	IF T1 ≠ 0 THEN BEGIN EMITV(0,T1,T2); P1 + 0 END;	15366000 T 0222
	END OF SUBSCRIPTED VARIABLES	15367000 T 0222
	ELSE	15368000 T 0225
		15369000 T 0225
		15370000 T 0226
		15376000 T 0229
		15376100 T 0229

```

        BEGIN COMMENT LABELID;
        T1:=TAKE(T2:=GIT(TALL));
        PUT(L,T2);
        IF T1=0 THEN T1:=L;
        IF (T1+L-T1) DIV 4 > 127 THEN BEGIN T1+0;FLAG(50);END;
        EMIT(T1*4+3);
        STEPIT;
        END OF LABELID;
EXIT :   END OF THE VARIABLE ROUTINE;

```

```

15376200 T 0229
15376300 T 0230
15376400 T 0232
15376500 T 0233
15376600 T 0235
15376700 T 0239
15376800 T 0241
15376900 T 0241
15377000 T 0241
102 IS 246 LONG, NEXT SEG 3

```

```

COMMENT THIS SECTION GENERATES CODE FOR STREAM PROCEDURES;
COMMENT DO LABEL DECS UPON APPEARANCE OF LABEL ;
PROCEDURE DECLARELABEL ;
PRT(711) = DECLARELABEL

```

```

        BEGIN
        KLASSE + STLABID;
        VONF + FORMALF + FALSE;
        ADDRSE + 0;
        MAKEUPACCUM; E; PUTNBUMP(0);
        ELBAT[I] + ACCUM[0]& LASTINFO[35:35:13];
        END;

```

```

16000000 T 0586
16000050 T 0586
16000100 T 0586

16000200 T 0586
16000300 T 0586
16000400 T 0587
16000500 T 0589
16000600 T 0589
16000700 T 0591
16000800 T 0594

```

PROCEDURE STREAMSTMT ;

```

        BEGIN
        DEFINE LFTPAREN=LEFTPAREN#,LOC=[36:12]#,LASTGT=[24:12]#,
                LOCFLD=36:36:12#,LGTFLD=24:24:12#;
        DEFINE LEVEL=LVL#,ADOP=ADOP#;
DEFINE
JFW = 39#, COMMENT 7.5.5.1 JUMP FORWARD UNCONDITIONAL ;
RCA = 40#, COMMENT 7.5.7.6 RECALL CONTROL ADDRESS ;
JRV = 47#, COMMENT 7.5.5.2 JUMP REVERSE UNCONDITIONAL ;
CRF = 35#, COMMENT 7.5.10.6 CALL REPEAT FIELD ;
BNS = 42#, COMMENT 7.5.5.5 BEGIN LOOP ;
NOP = 1#, COMMENT ;
ENS = 41#, COMMENT 7.5.5.6 END LOOP ;
TAN = 30#, COMMENT 7.5.3.7 TEST FOR ALPHAMERIC ;
BIT = 31#, COMMENT 7.5.3.8 TEST BIT ;
JFC = 37#, COMMENT 7.5.5.3 JUMP FORWARD CONDITIONAL ;
SED = 06#, COMMENT 7.5.7.8 SET DESTINATION ADDRESS ;
RSA = 43#, COMMENT 7.5.7.4 RECALL SOURCE ADDRESS ;
TRP = 60#, COMMENT 7.5.2.2 TRANSFER PROGRAM CHARACTERS ;
BSS = 3#, COMMENT 7.5.6.6 SKIP SOURCE BIT ;
BSD = 2#, COMMENT 7.5.6.5 SKIP DESTINATION BITS ;
SEC = 34#, COMMENT 7.5.10.1 SET COUNT ;
JNS = 38#, COMMENT 7.5.5.7 JUMP OUT LOOP ;

```

PROCEDURE ADJUST;

PRT(712) = ADJUST

```

16001000 T 0594
16002000 T 0594
16003000 T 0594
START OF SEGMENT ***** 103
16004000 T 0000
16005000 T 0000
16006000 T 0000
16007000 T 0000
16008000 T 0000
16009000 T 0000
16010000 T 0000
16011000 T 0000
16012000 T 0000
16013000 T 0000
16014000 T 0000
16015000 T 0000
16016000 T 0000
16017000 T 0000
16018000 T 0000
16019000 T 0000
16020000 T 0000
16021000 T 0000
16022000 T 0000
16023000 T 0000
16023100 T 0000

```

COMMENT FIXC EMITS BASICLY FORWARD JUMPS. HOWEVER IN THE CASE
OF INSTRUCTIONS INTERPTED AS JUMPS BECAUSE OF A CRF ON
A VALUE = 0 AND THE JUMP ≥ 64 SYLLABLES A JFW 1 AND
A RCA L (L IS STACK ADDRESS OF A PSEUDO LABEL WHICH
MUST ALSO BE MANUFACTURED) IS EMITTED. ;

PROCEDURE FIXC(S); VALUE S; REAL S;

PRT(713) = FIXC

BEGIN
REAL SAVL,D,F;

STACK(F+2) = SAVL
STACK(F+3) = D
STACK(F+4) = F

IF D+ (SAVL+L) = (L+S)-1 ≤ 63 THEN EMITC(D,GET(S))
ELSE FLAG(700);
L+SAVL ;
END FIXC ;

COMMENT EMITJUMP IS CALLED BY GOTOS AND JUMPCHAIN.
THIS ROUTINE WILL EMIT A JUMP IF THE DISTANCE IS ≤ 63
SYLLABLES ,OTHERWISE, IT GETS A PRT CELL AND STUFFS THE
STACK ADDRESS INTO THE LABEL ENTRY IN INFO AND EMITS AN
RCA ON THIS STACK CELL. AT EXECUTION TIME ACTUAL PARAPART
INSURES US THAT THIS CELL WILL CONATIN A LABEL DESCRIPTOR
POINTING TO OUR LABEL IN QUESTION. ;

PROCEDURE EMITJUMP(E); VALUE E; REAL E;

PRT(714) = EMITJUMP

BEGIN
REAL T,D;

STACK(F+2) = T
STACK(F+3) = D

STACK(F+4) = ADDR

REAL ADDR;
IF ABS(
D+(T+TAKE(GIT(E)).LOC)-L-1)≥64 THEN
FLAG(700)
ELSE EMITC(D,IF D < 0 THEN JRV ELSE JFW);
END EMIT JUMP;

COMMENT WHEN JUMPCHAIN IS CALLED THERE IS A LINKEDLIST IN THE CODE
ARRAY WHERE JFWs MUST BE PLACED. THE 1ST LINK IS POINTED
TO BY THE LOC FIELD OF EACH LABEL ENTRY IN INFO, THE LAST
LINK IS = 4096. ;

PROCEDURE JUMPCHAIN(E); VALUE E;REAL E;

PRT(715) = JUMPCHAIN

BEGIN
REAL SAVL ,LINK;

16024000 T 0000
16025000 T 0000
16026000 T 0000
16027000 T 0000
16028000 T 0000
16029000 T 0000

16030000 T 0000
16031000 T 0000

START OF SEGMENT ***** 104

16032000 T 0000
16033000 T 0004
16034000 T 0006
16057000 T 0007

104 IS 10 LONG, NEXT SEG 103

16058000 T 0000
16059000 T 0000
16060000 T 0000
16061000 T 0000
16062000 T 0000
16063000 T 0000
16064000 T 0000
16065000 T 0000

16066000 T 0000
16067000 T 0000

START OF SEGMENT ***** 105

16068000 T 0000
16069000 T 0000
16070000 T 0000
16071000 T 0004
16079000 T 0005
16080000 T 0009

105 IS 12 LONG, NEXT SEG 103

16081000 T 0000
16082000 T 0000
16083000 T 0000
16084000 T 0000
16085000 T 0000

16086000 T 0000
16087000 T 0000

START OF SEGMENT ***** 106

```
STACK(F+2) = SAVL
STACK(F+3) = LINK
```

```
SAVL ← L;
L ← TAKE(GIT(E)), LASTGT ;
WHILE L ≠ 4095 DO
  BEGIN
    LINK ← GET(L);
    EMITJUMP( E);
    L ← LINK
  END;
L ← SAVL;
END JUMPCHAIN ;
```

```
16088000 T 0000
16089000 T 0000
16090000 T 0003
16091000 T 0004
16092000 T 0004
16093000 T 0005
16094000 T 0006
16095000 T 0006
16096000 T 0009
16097000 T 0009
106 IS 12 LONG, NEXT SEG 103
```

COMMENT NESTS COMPILES THE NEST STATEMENT.
 A VARIABLE NEST INDEX CAUSES THE CODE,
 CRF V, BNS 0 ,NOP,NOP, TO BE GENERATED INITIALLY.
 AT THE RIGHT PAREN THE BNS IS FIXED WITH THE LENGTH OF
 THE NEST (NUMBER OF SYLLABLES) IF THE LENGTH ≤63, OTHERWISE
 IT IS FIXED WITH A 1 AND THE NOPS REPLACED WITH JFW 1,
 RCA P. THIS IS DONE BECAUSE THE VALUE OF V AT EXECUTION
 MAY = 0 AND THIS CODE CAUSES A JUMP AROUND THE NEST.
 JUMPOUT INFO IS REMEMBERED IN A RECURSIVE CELL AND
 NEST LEVEL INCREASED BY ONE.
 WHEN THE RIGHT PAREN IS REACHED, (IF THE STATEMENTS IN
 THE NEST COMPILED), JOINFO IS CHECKED FOR THE EXISTANCE
 OF JUMPOUT STATEMENTS IN THE NEST, IF SO, THE THE JUMPS
 ARE FIXED BY FAKING TOTOS INTO COMPILING THE REQUIRED
 JUMPS.
 FINALLY THE BNS IS FIXED, IF REQUIRED, AND NEST LEVEL
 AND JOINFO RESTORED TO THEIR ORIGINAL VALUES. ;

```
16098000 T 0000
16099000 T 0000
16100000 T 0000
16101000 T 0000
16102000 T 0000
16103000 T 0000
16104000 T 0000
16105000 T 0000
16106000 T 0000
16107000 T 0000
16108000 T 0000
16109000 T 0000
16110000 T 0000
16111000 T 0000
16112000 T 0000
16113000 T 0000
16114000 T 0000
16115000 T 0000
```

PROCEDURE NESTS;

PRT(716) = NESTS

```
BEGIN
LABEL EXIT;
REAL JOINT, BNSFIX;
```

```
STACK(F+2) = JOINT
STACK(F+3) = BNSFIX
```

```
IF ELCLASS ≠ LITNU THEN
  BEGIN
    EMITC(ELBAT[I], ADDRESS, CRF); BNSFIX ← L;
    EMIT(BNS);
  END
ELSE EMITC(ELBAT[I], ADDRESS, BNS);
IF STEPI ≠ LFTPAREN THEN BEGIN ERR(262); GO TO EXIT END;
NESTLEVEL ← NESTLEVEL + 1;
JOINT ← JOINFO;
JOINFO ← 0;
DO BEGIN
  STEPIT; ERRORTOG ← TRUE; STREAMSTMT
  END UNTIL ELCLASS ≠ SEMICOLON ;
IF ELCLASS ≠ RTPAREN THEN BEGIN ERR(262); GO TO EXIT END;
EMIT ( ENS);
```

```
16116000 T 0000
16117000 T 0000
START OF SEGMENT ***** 107
16118000 T 0000
```

```
16119000 T 0000
16120000 T 0000
16121000 T 0001
16122000 T 0003
16123000 T 0004
16124000 T 0004
16125000 T 0006
16126000 T 0009
16127000 T 0010
16128000 T 0011
16129000 T 0012
16130000 T 0013
16131000 T 0014
16132000 T 0016
16133000 T 0018
```

```

      IF JOINFO ≠ 0 THEN
        BEGIN
      COMMENT  PREPAKE TO CALL JUMPCHAIN FORJUMPOUTS;
        ADJUST;
        PUT(TAKE(GIT(JOINFO))&L[LOCFLD],GIT(JOINFO));
        JUMPCHAIN(TAKE(JOINFO)&JOINFO[35:35:13]);
        END;
      IF BNSFIX ≠ 0 THEN FIXC(BNSFIX);
      NESTLEVEL + NESTLEVEL-1;
      JOINFO + JOINT ;
EXIT: END  NESTS  ;

```

```

16134000 T 0019
16135000 T 0020
16136000 T 0020
16137000 T 0020
16138000 T 0021
16139000 T 0024
16140000 T 0026
16141000 T 0026
16142000 T 0028
16143000 T 0030
16144000 T 0030
107 IS 34 LONG, NEXT SEG 103

```

```

      COMMENT  LABELS HANDLES STREAM LABELS.
      ALL LABELS ARE ADJUSTED TO THE BEGINING OF THE NEXT
      WORD (IN THE PROGRAMSTREAM).
      IF A GO TO HAS NOT BEEN ENCOUNTERED BEFORE THE LABEL
      THEN THE NEST LEVEL FIELD IS ENTERED AND THE DEFINED BIT,
      [1:1], SET TO ONE, FOR DEFINED LABELS, IF WHERE A GO TO
      HAS APPEARED, A CHECK IS MADE THAT THE CURRENT NEST LEVEL
      MATCHES THE LEVEL OF THE LABEL.
      MULTIPLE OCCURANCES ARE ALSO CHECKED FOR AND FLAGGED.
      FINALLY, JUMPCHAIN IS CALLED TO FIX UP ANY FORWARD GO TOS
      AND GET A PRT LOCATION FOR ANY JUMPS ≥ 64 SYLLABLES. ;

```

```

16145000 T 0000
16146000 T 0000
16147000 T 0000
16148000 T 0000
16149000 T 0000
16150000 T 0000
16151000 T 0000
16152000 T 0000
16153000 T 0000
16154000 T 0000
16155000 T 0000
16156000 T 0000

```

```

      PROCEDURE LABELS;
PRT(717) = LABELS

```

```

      BEGIN
      REAL GT1;

```

```

16157000 T 0000
16157100 T 0000

```

```

START OF SEGMENT ***** 108

```

```

STACK(F+2) = GT1

```

```

      ADJUST;
      GT1 + ELBAT[1];
      IF STEP1 ≠ COLON THEN EHR(258)
      ELSE
        BEGIN
          IF TAKE(GT2+GIT(GT1)).LOC ≠ 0 THEN FLAG(259);
          IF GT1>0 THEN
            BEGIN
              PUT(=(TAKE(GT1)&NESTLEVEL[11:43:5]),GT1);
              PUT(=L,GT2)
            END
          ELSE
            BEGIN
              IF GT1.LEVEL≠NESTLEVEL THEN FLAG(257);
              PUT(=(L)&TAKE(GT2)[LGTFLD],GT2);
              JUMPCHAIN(GT1);
            END;
          END
        ; STEP1;
      END LABELS ;

```

```

16158000 T 0000
16159000 T 0000
16160000 T 0001
16161000 T 0003
16162000 T 0003
16163000 T 0004
16164000 T 0008
16165000 T 0009
16166000 T 0009
16167000 T 0012
16168000 T 0013
16169000 T 0013
16170000 T 0013
16171000 T 0014
16172000 T 0016
16173000 T 0019
16174000 T 0020
16175000 T 0020
16176000 T 0020
16177000 T 0020

```

```

108 IS 23 LONG, NEXT SEG 103

```

COMMENT IFS COMPILES IF STATEMENTS,
 FIRST THE TEST IS COMPILED, NOTE THAT IN THE
 CONSTRUCTS "SC RELOP DC" AND "SC RELOP STRING" THAT
 THE SYLLABLE EMITTED IS FETCHED FROM ONE OF TWO FIELDS
 IN THE ELBAT WORD FOR THE RELATIONAL OPERATOR. OTHERWISE
 THE CODE IS EMITTED STRAIGHTAWAY.
 A TEST IS MADE TO SEE WHETHER THE STATEMENT AFTER THE
 "THEN" COULD POSSIBLY BE LONGER THAN 63 SYLLABLES, AND IF
 SO, Z NOPS ARE EMITTED FOR FIXC IN CASE A RCA WILL HAVE
 TO BE GENERATED.
 THIS PROCEDURE DOES NO OPTIMAZATION IN THE CASES
 IF THEN GO TO L, IF THEN STATEMENT ELSE GO TO L, OR
 IF THEN GO TO L1 ELSE GO TO L2 ;

PRT(720) = IFS

PROCEDURE IFS; BEGIN

DEFINE COMPARECODE=[42:6]#,TESTCODE=[36:6]#;

LABEL IFSB,IFTOG,IFSC,EXIT;
 SWITCH IFSW + IFSB,IFTOG,IFSC;
 REAL ADDR,FIX1,FIX2 ;

STACK(F+2) = ADDR
 STACK(F+3) = FIX1
 STACK(F+4) = FIX2

ADDR+1 ;
 GO TO IFSW[STEP1 = SBV+1] ;
 IF ELCLASS=LOCLID THEN
 BEGIN
 EMITC(ELBAT[I],ADDRESS,CRF);
 ADDR+0;
 END
 ELSE
 IF ELCLASS=LITNO THEN ADDR + ELBAT[I],ADDRESS
 ELSE BEGIN ERR(250); GO TO EXIT END;
 IF STEP1 ≠ SCV THEN BEGIN ERR(263);GO TO EXIT END;
 IFSC: IF STEP1 ≠ RELOP THEN BEGIN ERR(264);GO TO EXIT END;
 IF STEP1 = DCV THEN EMITC(ADDR,ELBAT[I-1],COMPARECODE)
 ELSE
 IF ELCLASS = STRNGCON THEN
 EMITC(ACCUM[1],[18:6],ELBAT[I-1],TESTCODE)
 ELSE
 IF ELCLASS=LITNO THEN EMITC(C,ELBAT[I-1],TESTCODE) ELSE
 IF ELCLASS\$IDMAX AND Q="SALPHA" THEN EMITC(17,TAN)
 ELSE BEGIN ERR(265);GO TO EXIT END;
 GO TO IFTOG ;
 IFSB: EMITC(1,BIT);
 IFTOG: IF STEP1 ≠ THENV THEN BEGIN ERR(266); GO TO EXIT END;
 FIX1 + L;
 EMIT(JFC);
 IF STEP1≠ELSEV THEN
 STREAMSTMT;
 IF ELCLASS= ELSEV THEN
 BEGIN
 FIX2 + L; EMIT(JFW);
 FIXC(FIX1);
 STEPIT;
 STREAMSTMT;
 FIXC(FIX2);

16178000 T 0000
 16179000 T 0000
 16180000 T 0000
 16181000 T 0000
 16182000 T 0000
 16183000 T 0000
 16184000 T 0000
 16185000 T 0000
 16186000 T 0000
 16187000 T 0000
 16188000 T 0000
 16189000 T 0000
 16190000 T 0000
 16191000 T 0000

16192000 T 0000
 START OF SEGMENT ***** 109
 16193000 T 0000
 16194000 T 0000
 16195000 T 0004

16196000 T 0004
 16197000 T 0005
 16198000 T 0009
 16199000 T 0009
 16200000 T 0010
 16201000 T 0012
 16202000 T 0012
 16203000 T 0012
 16204000 T 0012
 16205000 T 0014
 16206000 T 0017
 16207000 T 0020
 16208000 T 0023
 16209000 T 0026
 16210000 T 0027
 16211000 T 0028
 16212000 T 0031
 16212500 T 0032
 16213000 T 0036
 16214000 T 0039
 16215000 T 0043
 16216000 T 0043
 16217000 T 0045
 16218000 T 0047
 16219000 T 0048
 16220000 T 0049
 16229000 T 0050
 16230000 T 0051
 16231000 T 0052
 16232000 T 0052
 16233000 T 0054
 16234000 T 0054
 16235000 T 0055
 16236000 T 0055


```

END
ELSE FIXC(FIX1);
EXIT:END IFS ;

```

```

16237000 T 0056
16238000 T 0056
16239000 T 0057
109 IS 61 LONG, NEXT SEG 103

```

```

COMMENT GOTOS HANDLES GO TO AND THE LAST PART OF JUMP OUT TO
STATEMENTS.
IF THE LABEL HAS BEEN ENCOUNTERED THEN EMITJUMP IS CALLED
AN PRODUCES A JRV OR RCA IN THE CASE OF JUMPS264 SYLLABL
ES. OTHERWISE, A LINK IS EMITTED POINTING ANY PREVIOUS
GO TOS IN THE CASE OF FORWARD JUMPS.
FINALLY, IF THE NEST LEVEL IS DEFINED THEN IT IS CHECKED
AGAINST THE CURRENT LEVEL MINUS THE NUMBER OF LEVELS TO
BE JUMPED OUT. OTHERWISE, NEST LEVEL IS DEFINED. ;

```

```

16240000 T 0000
16241000 T 0000
16242000 T 0000
16243000 T 0000
16244000 T 0000
16245000 T 0000
16246000 T 0000
16247000 T 0000
16248000 T 0000
16249000 T 0000

```

```

PRT(721) = GOTOS

```

```

PROCEDURE GOTOS;

```

```

BEGIN
LABEL EXIT;

IF STEPI # TOV THEN I+I=1 ;
IF STEPI # STLABID THEN IF ELCLASS ≤ IDMAX THEN
    DECLARE LABEL ELSE BEGIN ERR(260); GU TO EXIT END;
IF (GT2+TAKE(GIT(GT1+ELBAT[I]))).MON=1
OR GT2.LOC#0 THEN EMITJUMP(GT1)
ELSE
    BEGIN PUT(0&L[24:36:12],GIT(GT1));
        IF GT1>0 THEN
            BEGIN
                PUT(=(TAKE(GT1)&(NESTLEVEL-JUMPLEVEL))[11:43:5]),GT1);
                EMITN(1023);
            END
        ELSE
            BEGIN
                IF GT1,LEVEL # NESTLEVEL-JUMPLEVEL THEN FLAG(257);
                EMIT(GT2.LASTGT);
            END
        END;
    JUMPLEVEL+0 ;
EXIT: END GOTOS ;

```

```

16250000 T 0000
16251000 T 0000
START OF SEGMENT ***** 110
16252000 T 0000
16253000 T 0002
16253100 T 0005
16254000 T 0007
16255000 T 0010
16256000 T 0013
16257000 T 0014
16258000 T 0017
16259000 T 0017
16260000 T 0018
16261000 T 0021
16262000 T 0022
16263000 T 0022
16264000 T 0022
16265000 T 0022
16266000 T 0025
16267000 T 0027
16268000 T 0027
16269000 T 0027
16270000 T 0027
110 IS 29 LONG, NEXT SEG 103

```

```

COMMENT RELEASES COMPILES THE STREAM RELEASE STATEMENT.
THE CODE GENERATED IS :
    SED FILE
    RSA 0.
AT EXECUTION TIME THIS CAUSES AN INVALID ADDRESS WHICH IS
INTERPETED BY THE MCP TO MEAN RELEASE THE FILE POINTED TO
BY THE DESTINATION ADDRESS.
THE MONITOR BIT IS SET IN INFO FOR THE LOCAL VARIABLE SO
THAT ACUTAL PARAPART MAY BE INFORMED LATER THAT A FILE
MUST BE PASSED FOR THIS FORMAL PARAMETER;

```

```

16271000 T 0000
16272000 T 0000
16273000 T 0000
16274000 T 0000
16275000 T 0000
16276000 T 0000
16277000 T 0000
16278000 T 0000
16279000 T 0000
16280000 T 0000

```

COMMENT INDEXS COMPILE STATEMENTS BEGINING WITH SI,DI,CI,TALLY
OR LOCALIDS .
THREE CASES PRESENT THEMSELVES,
LETING X BE EITHER OF SI,DI,CI OR TALLY, THEY ARE:
CASE I LOCLID + X
CASE II X + X ...
CASE III X + EITHER LOC,LOCLID,SC OR DC.
THE VARIABLE "INDEX" IS COMPUTED,DEPENDING UPON WHICH
CASE EXISTS,SUCH THAT ARRAY ELEMENT "MACRO[INDEX]"CONTAINS
THE CODE TO BE EMITTED.
EACH ELEMENT OF MACRO HAS 1-3 SYLLABLES ORDERED FROM
RIGHT TO LEFT, UNUSED SYLLABLES MUST = 0. EACH MACRO
MAY REQUIRE AT MOST ONE REPEAT PART.
IN THIS PROCEDURE,INDEXS,THE VARIABLE "ADDR" CONTAINS THE
PROPER REPEAT PART BY THE TIME THE LABEL "GENERATE" IS
ENCOUNTERED, THE SYLLABLES ARE FETCHED FROM MACRO[TYPE]
ONE AT A TIME AND IF THE REPEAT PART ≠ 0 THEN"ADDR" IS
USED AS THE REPEAT PART,THUS BUILDING A SYLLABLE WITH
THE PROPER ADDRESS AND OPERATOR .
NOTE: IF MACRO[TYPE] = 0 THEN THIS SIGNIFIES A SYNTAX
ERROR.

PROCEDURE INDEXS;

PRT(722) = INDEXS

BEGIN
LABEL EXIT,GENERATE,L,L1;

INTEGER TCLASS,INDEX,ADDR,J;

STACK(F+2) = TCLASS
STACK(F+3) = INDEX
STACK(F+4) = ADDR
STACK(F+5) = J

TCLASS = ELCLASS ;
IF STEPI ≠ ASSIGNOP THEN BEGIN ERR(251); GO TO EXIT END;
IF TCLASS = LOCLID THEN
BEGIN
IF SIV>STEPI OR ELCLASS>TALLYV THEN GO TO L;
INDEX = 32 + ELCLASS-SIV;
ADDR = ELBAT[I-2],ADDRESS;
GO TO GENERATE;
END;
IF TCLASS = STEPI THEN
BEGIN
IF STEPI ≠ ADDOP OR STEPI ≠ LITNO AND ELCLASS ≠ LOCLID THEN
GO TO L;
INDEX = TCLASS-SIV
+REAL(ELBAT[I-1],ADDRESS-SUB) × 4
+ REAL(ELCLASS =LOCLID) × 8;
END

16281000 T 0000
16282000 T 0000
16283000 T 0000
16284000 T 0000
16285000 T 0000
16286000 T 0000
16287000 T 0000
16288000 T 0000
16289000 T 0000
16290000 T 0000
16291000 T 0000
16292000 T 0000
16293000 T 0000
16294000 T 0000
16295000 T 0000
16296000 T 0000
16297000 T 0000
16298000 T 0000
16299000 T 0000
16300000 T 0000
16301000 T 0000
16302000 T 0000
16303000 T 0000
16304000 T 0000
16305000 T 0000
16306000 T 0000
16307000 T 0000
16308000 T 0000
16309000 T 0000
16310000 T 0000
16311000 T 0000

16312000 T 0000
16313000 T 0000
START OF SEGMENT ***** 111
16314000 T 0000

16315000 T 0000
16316000 T 0000
16317000 T 0003
16318000 T 0004
16319000 T 0004
16320000 T 0007
16321000 T 0009
16322000 T 0011
16323000 T 0011
16324000 T 0011
16325000 T 0012
16326000 T 0013
16327000 T 0015
16328000 T 0017
16329000 T 0017
16330000 T 0020
16331000 T 0022

```

ELSE
  BEGIN
    INDEX ← TCLASS -SIV
    + ( IF ELCLASS = LOCLID THEN 16 ELSE
      IF ELCLASS = LOCV THEN 20 ELSE
      IF ELCLASS = SCV THEN 24 ELSE
      IF ELCLASS= DCV THEN 28 ELSE 25));
    IF ELCLASS = LOCV THEN
      IF STEPI ≠ LOCLID THEN GO TO L;
    IF ELCLASS = LITNO AND TCLASS = TALLYV THEN
      BEGIN EMITC(ELBAT[I],ADDRESS,SEC);GO TO EXIT END;
    END ;
    ADDR ← ELBAT[I],ADDRESS;
  GENERATE;
    IF MACRO[INDEX]= 0 THEN
      L: BEGIN ERR(250);GO TO EXIT END;
      J ← 8; TCLASS ←0 ;
    L1: MOVECHARACTERS(2,MACRO[INDEX],J+J-2,TCLASS,6 );
    IF TCLASS≠0 THEN
      BEGIN
        EMITC(IF TCLASS≥64 THEN ADDR ELSE 0,TCLASS);
        GO TO L1
      END;
    EXIT;END INDEXS ;

```

```

16332000 T 0022
16333000 T 0022
16334000 T 0023
16335000 T 0023
16336000 T 0026
16337000 T 0028
16338000 T 0030
16339000 T 0033
16340000 T 0033
16341000 T 0035
16342000 T 0037
16343000 T 0040
16344000 T 0040
16345000 T 0041
16346000 T 0042
16347000 T 0043
16348000 T 0045
16349000 T 0046
16350000 T 0050
16351000 T 0050
16352000 T 0051
16353000 T 0054
16354000 T 0054
16355000 T 0054
111 IS 58 LONG, NEXT SEG 103

```

```

COMMENT DSS COMPILES DESTINATION STREAM STATEMENTS.
DS← LIT"STRING" IS HANDLED AS A SPECIAL CASE BECAUSE THE
STRING MUST BE SCANNED FROM RIGHT TO LEFT,REPEATEDLY IF
NECESSARY, AND EMITTED TO THE PROGRAM STREAM. IN
ALL OTHER CASES,THE ELBAT WORD CONTAINS THE OPERATOR IN
THE OPCODE FIELD ;

```

PROCEDURE DSS;

PRT(723) = DSS

```

BEGIN
  INTEGER ADDR,J,K,L,T;

```

```

STACK(F+2) = ADDR
STACK(F+3) = J
STACK(F+4) = K
STACK(F+5) = L
STACK(F+6) = T

```

```

LABEL EXIT,L1)
DEFINE OPCODE=[2716]#;
IF STEPI ≠ ASSIGNOP THEN BEGIN ERR(251); GO TO EXIT END;
IF STEPI = LOCLID THEN
  BEGIN
    EMITC(ELBAT[I],ADDRESS,CRF);
    ADDR← 0;
    IF STEPI = LITV THEN GO TO L1
  END
ELSE IF ELCLASS= LITNO THEN
  BEGIN
    ADDR ← ELBAT[I],ADDRESS; STEPIT ;

```

```

16356000 T 0000
16357000 T 0000
16358000 T 0000
16359000 T 0000
16360000 T 0000
16361000 T 0000
16362000 T 0000

16363000 T 0000
16364000 T 0000
START OF SEGMENT ***** 112

```

```

16365000 T 0000
16366000 T 0000
16367000 T 0000
16368000 T 0002
16369000 T 0003
16370000 T 0004
16371000 T 0006
16372000 T 0006
16373000 T 0007
16374000 T 0008
16375000 T 0009
16376000 T 0010

```

```

        END
    ELSE ADDR ← 1 ;
    IF Q = "4FILL" THEN EMITC(ADDR,10) ELSE
    IF ELCLASS = TRANSFER THEN EMITC(ADDR,ELBAT[I],UPCODE)
    ELSE
    IF ELCLASS = LITV THEN
        BEGIN
            EMITC(ADDR,TRP);
            IF STEPI≠STRNGCON THEN
                BEGIN ERR(255);GO TO EXIT END;
            IF ADDR MOD 2 ≠ 0 THEN
                BEGIN
                    EMIT(ACCUM[I],[18:6]); J ← 1;
                END ;
            FOR K ←J+2 STEP 2 UNTIL ADDR DO
                BEGIN
                    FOR L ←6,7 DO
                        MOVECHARACTERS(1,ACCUM[I],2+(IF J+J+1>COUNT THEN J+1
STACK(F+7) = *TEMPORARY STORAGE*
                                ELSE J),T,L );
                                EMIT(T);
                            END
                        END
                    ELSE
                        L1: ERR(250);
                    EXIT;END DSS ;

```

```

16377000 T 0012
16378000 T 0012
16378500 T 0013
16379000 T 0015
16380000 T 0019
16381000 T 0020
16382000 T 0021
16383000 T 0021
16384000 T 0022
16384500 T 0023
16385000 T 0025
16386000 T 0026
16387000 T 0027
16388000 T 0029
16389000 T 0029
16390000 T 0033
16391000 T 0033
16392000 T 0038
16393000 T 0041
16394000 T 0044
16395000 T 0045
16396000 T 0046
16397000 T 0046
16398000 T 0047
112 IS 52 LONG, NEXT SEG 103

```

```

COMMENT SKIPS COMPILES THE SKIP BIT STATEMENT.
    IF THE REPEAT INDEX IS A LOCALID THEN A CRF IS EMITTED,
    A BSS OR BSD IS THEN EMITTED FOR SKIP SOURCE BITS (SB)
    OR SKIP DESTINATION BITS (DB) RESPECTIVELY ;
PROCEDURE SKIPS ;
PRT(724) = SKIPS
    BEGIN
    REAL ADDR;
STACK(F+2) = ADDR
    IF STEPI = LOCLID THEN
        BEGIN
            EMITC(ELBAT[I],ADDRESS,CRF); ADDR←0; STEPIT;
        END
    ELSE IF ELCLASS = LITNO THEN
        BEGIN
            ADDR← ELBAT[I],ADDRESS; STEPIT
        END
    ELSE ADDR ← 1 ;
    IF ELCLASS =SBV THEN EMITC(ADDR,BSS)
    ELSE
    IF ELCLASS =DBV THEN EMITC(ADDR,BSD)
    ELSE ERR(250);
    END SKIPS ;

```

```

16399000 T 0000
16400000 T 0000
16401000 T 0000
16402000 T 0000
16403000 T 0000
16404000 T 0000
16405000 T 0000
START OF SEGMENT ***** 113
16406000 T 0000
16407000 T 0001
16408000 T 0001
16409000 T 0004
16410000 T 0004
16411000 T 0005
16412000 T 0006
16413000 T 0007
16414000 T 0008
16415000 T 0009
16416000 T 0011
16417000 T 0011
16418000 T 0014
16419000 T 0015
113 IS 18 LONG, NEXT SEG 103

```

COMMENT JUMPS COMPILES JUMP OUT AND JUMP OUT TO STATEMENTS,
 JUMP OUT TO STATEMENTS CAUSE JUMP LEVEL TO BE SET TO
 THE NUMBER OF LEVELS SPECIFIED, THEN THIS NUMBER OF
 JNS ARE EMITTED AND GOTOS IS CALLED TO COMPILE THE
 JUMP INSTRUCTION.
 SIMPLE JUMP OUTS ARE HANDLED BY EMITTING ONE JNS, ENTERING
 A PSEUDO STLABID IN INFO AND SETTING ELBAT[I] SUCH THAT
 THE GOTOS PROCEDURE WILL PERFORM THE ACTION OF SETTING
 UP THE LINKS FOR LATER FIX UPS. THE NEXT STATEMENT CAUSES
 THESE FIX UPS (IF EMITTING OF JUMP INSTRUCTIONS) BY CALLING
 GO TOS WHEN THE RIGHT PAREN IS ENCOUNTERED. ;

PROCEDURE JUMPS;

PRI(725) = JUMPS

```

BEGIN
  JUMPLEVEL+1;
  IF STEPI#DECLARATORS THEN IF ACCUM[1]#"3OUT00" THEN
    FLAG(261);
  IF STEPI = LITNU THEN JUMPLEVEL+ ELBAT[I],ADDRESS
  ELSE BEGIN
    IF ELCLASS# TOV AND ELCLASS# STLABID THEN
      BEGIN
        COMMENT SIMPLE JUMP OUT STATEMENT;
        IF JOINFO = 0 THEN
          BEGIN
            JOINFO = NEXTINFO ;
            PUTNBUMP(STACKHEAD[0].LINK&(STLABID*2+1)
              [214018]&2[2714018 ]));
            PUTNBUMP(0&(JOINFO-LASTINFO ) [ 414018]);
            PUTNBUMP (0);
            LASTINFO = JOINFO;
          END;
          ELBAT[I+ I-1]+ TAKE(JOINFO)&JOINFO[35135113];
          END; I+I-1 ;
        END;
        FOR GT1+ 1 STEP 1 UNTIL JUMPLEVEL DO
          EMIT( JNS);
        GOTOS;
      END JUMPS;

```

```

16420000 T 0000
16421000 T 0000
16422000 T 0000
16423000 T 0000
16424000 T 0000
16425000 T 0000
16426000 T 0000
16427000 T 0000
16428000 T 0000
16429000 T 0000
16430000 T 0000
16431000 T 0000

```

```

16432000 T 0000
16433000 T 0000
16434000 T 0001
16434100 T 0004
16435000 T 0005
16436000 T 0007
16437000 T 0010
16438000 T 0011
16439000 T 0012
16440000 T 0012
16441000 T 0013
16442000 T 0013
16443000 T 0014
16444000 T 0016
16445000 T 0018
16446000 T 0021
16447000 T 0021
16448000 T 0022
16449000 T 0022
16450000 T 0026
16451000 T 0027
16452000 T 0027
16453000 T 0029
16454000 T 0032
16455000 T 0032

```

COMMENT STREAMSTMT ENVOKES THE APPROPRIATE PROCEDURE TO HANDLE
 THE VARIOUS AND SUNDRY STREAM PROCEDURE STATEMENTS.
 THE STATEMENTS ARE BROKEN DOWN AS FOLLOWS:

IDENTIFIED BY	PROCEDURE ENVOKED
END	GO TO FINI
SEMICOLON	GO TO FINI
)	GO TO FINI
IF	IFS
GO	GOTOS
RELEASE	RELEASES
BEGIN	COMPOUNDTAIL
SI,DI,CI,TALLY,LOCALID INDEXS	
DS	DSS
SKIP	SKIPS
JUMP	JUMPS

```

16456000 T 0032
16457000 T 0032
16458000 T 0032
16459000 T 0032
16460000 T 0032
16461000 T 0032
16462000 T 0032
16463000 T 0032
16464000 T 0032
16465000 T 0032
16466000 T 0032
16467000 T 0032
16468000 T 0032
16469000 T 0032
16470000 T 0032

```

```

        LABELID          LABELS
        LITERAL NO.,LOCALID( NESTS
UPON EXITING,STREAMSTMT ASSURES THAT "I" POINTS TO
THE SEMICOLON ,END OR ) IN SYNTACTICALLY CORRECT PROGRAMS;
LABEL L,L1,L2,L3,L4,L5,L6,L7,L8,L9,L10,EXIT,FINI,START;
SWITCH TYPE + FINI,L,FINI,L3,L4,L5,L6,L7,L7,L7,L7,L8,L9,L10;
START:  GO TO TYPE[ ELCLASS=ENDV+1];
        IF ELCLASS= RTPAREN THEN GO TO FINI ;
        IF ELCLASS= STLABID THEN GO TO L2 ;

        IF ELCLASS <IDMAX AND ELCLASS#LOCLID THEN BEGIN
            DECLARELABEL; GO TO L2; END;
        IF ELCLASS = LITNO OR ELCLASS = LOCLID AND TABLE(I+1)
            = LFTPAREN THEN GO TO L1;
        IF ELCLASS = LOCLID THEN GO TO L7;
L1:      ERR( 250 ); GO TO FINI ;
L1:      NESTS;      GO TO EXIT;
L2:      LABELS;     GO TO START;
L3:      IFS;        GO TO FINI;
L4:      GOTOS;      GO TO EXIT;
L5:
L6:      I+I+1 ; COMPOUNDTAIL; GO TO FINI;
L7:      INDEXS;     GO TO EXIT;
L8:      DSS;        GO TO EXIT;
L9:      SKIPS;      GO TO EXIT;
L10:     JUMPS;      GO TO EXIT;
EXIT:    STEPIT;
FINI:    END    STREAMSTMT;

```

```

16471000 T 0032
16472000 T 0032
16473000 T 0032
16474000 T 0032
16475000 T 0032
16476000 T 0032
16477000 T 0043
16478000 T 0047
16481000 T 0048
16482000 T 0049
16482100 T 0049
16482200 T 0051
16482300 T 0052
16482400 T 0055
16482500 T 0057
16483000 T 0058
16484000 T 0060
16485000 T 0062
16486000 T 0063
16487000 T 0064
16488000 T 0065
16489000 T 0065
16490000 T 0067
16491000 T 0069
16492000 T 0070
16493000 T 0071
16494000 T 0072
16495000 T 0072

```

103 IS 74 LONG, NEXT SEG 3

```

MOVE(1,I,CODE(0));
TIME1 + TIME(1); PROGRAM;
ENDOFITALL;END MAIN BLOCK
END.

```

PRT(726) = *SEGMENT DESCRIPTOR*

```

16495100 T 0594
16495200 T 0599
16495210 T 0600
16495300 T 0601

```

3 IS 604 LONG, NEXT SEG 2
2 IS 41 LONG, NEXT SEG 1

```

PRT(527) = EXP INTRINSIC, SEGMENT NUMBER = 114.
PRT(526) = LN INTRINSIC, SEGMENT NUMBER = 115.
PRT(343) = OUTPUT(W) INTRINSIC, SEGMENT NUMBER = 116.
PRT(5) = BLOCK CONTROL INTRINSIC, SEGMENT NUMBER = 117.
PRT(530) = X TO THE I INTRINSIC, SEGMENT NUMBER = 118.
PRT(373) = GO TO SOLVER INTRINSIC, SEGMENT NUMBER = 119.
PRT(14) = ALGOL WRITE INTRINSIC, SEGMENT NUMBER = 120.
PRT(15) = ALGOL READ INTRINSIC, SEGMENT NUMBER = 121.
PRT(16) = ALGOL SELECT INTRINSIC, SEGMENT NUMBER = 122.
PRT(411) = DYNAMIC DIALS INTRINSIC, SEGMENT NUMBER = 123.
PRT(201) = FILE ATTRBUTS INTRINSIC, SEGMENT NUMBER = 124.

```

1 IS 2 LONG, NEXT SEG 0
125 IS 69 LONG, NEXT SEG 0

NUMBER OF ERRORS DETECTED = 0. COMPILATION TIME = 816 SECONDS.

PRT SIZE = 471; TOTAL SEGMENT SIZE = 7276 WORDS; DISK SIZE = 360 SEGS; NO. PGM. SEGS = 125

ESTIMATED CORE STORAGE REQUIRED = 13928 WORDS.

ESTIMATED AUXILIARY MEMORY REQUIRED = 0 WORDS.

NUMBER OF CARD-IMAGES PROCESSED = 6918.

A
0003 *02062500*
02063500

A
0003 *02054000*
02055000

A
0003 *02022000*
02023000

A
0003 *03050000*

A
0003 *04311000*
04311050 04311060 04315000 *04315000*

A
0003 *03049000*

A
0003 *03047100*

A
0003 *04500000*
04529000 04534000 04542000 04548000 04550000 04551000 04556000

A
0012 *02264100*
02265100 02265150 02265250 02265300 02265600 02265650 02265700 02265750 02265850 02265900 02265950
02266000

A
0036 *05130000*
05131000 05132000

A
0035 *05065000*
05065000 05066000

A
0041 *05334100*
05334100 05411100

A
0043 *05415000*
05417000 05419000 05421000 05422000

A
0043 *05413000*
05426000 05427000 05431000 05432000

A
0065 *08015000*
08181000 08188000

A
0065 *08027000*
*08035000**08037000*

A
0065 *08020000*
08021000

A
0089 *13359000*
13364000 13365000 13366000

A
0092 *13673100*
13673100

ABS

0035 *05065000*
05066000 05092000

ACC

0063 *07653500*
07654500 08027000 08083000 08084000

ACCUM

0003 *01304000*
02128500 02251000 02255000 02259000 02278000 02280000 02327000 02366100 02371000 02499000 02502000
02581000 02590000 02592000 02602600 02647000 02650000 02652000 02686000 02686500 02696000 02698000
02700000 02705000 02754000 02822000 02846000 02865000 02881000 02958500 05043000 05044000 05325460
07086200 07086300 07668500 07670000 07671000 10264700 10264900 10269000 10274000 12117000 12118000
12119000 12120000 12122000 13281000 13282000 13307000 13308000 13309000 13310000 13372000 13755000
14259000 14259080 14440000 14441000 16000700 16211000 16387000 16392000 16434000

ACCUM

0003 *02041000*
02043000

ACCUM

0007 *02089500*
02122500

ACCUM

0034 *05016000*
05028000

ACCUM

0054 *07038100*
07038500 07536000 07537000

ACCUM

0083 *10236000*
10239000

ACCLASS

0054 *07038000*

07041000 07047000 07057000 07069000 *07081000* 07089000

ACTUALPARAPART

0003 *07035000*

07410000 07438000

ADD

0003 *01633000*

05226000 07525000 08126000 08214000 13221000

ADD

0087 *13221000*

13248000 13250000 13252000 13252500 *13259000*

ADDC

0003 *13212000*

ADDITIONAL

0062 *07595000*

07601000 07603000 07610000

ADDDP

0103 *16005000*

16326000

ADDR

0105 *16068000*

ADDR

0109 *16195000*

*16196000**16201000**16204000* 16208000

ADDR

0113 *16405000*

*16408000**16412000**16414000* 16415000 16417000

ADDR

0112 *16364000*

*16371000**16376000**16378000* 16378500 16379000 16383000 16385000 16389000

ADDR

0111 *16314000*

*16321000**16344000* 16352000

ADDRES

0065 *08010000*

08055000 08067000 08068000 08075000 08179000 08212000

ADDRESS

0003 *01152000*

02850000	02896000	02900000	02923000	04508000	05131000	05217000	05233000	05280000	05281000	05344570
05344615	05344620	05344660	05344700	05344710	06016000	06072000	06108000	06185000	06324000	06325000
06326000	07071000	07090000	07393000	07439000	07745000	07768100	07994000	07995000	08021000	08029000
08044000	08055000	10277000	12015000	12024000	12036000	12047000	12108000	13205000	13248000	13259000
13394000	14085000	*14115000*	14129000	14349000	14399000	14430000	14437000	14461200	14483200	15094000
15097000	15106100	15107000	15110200	15110300	15233010	15233011	15233014	15267000	15276010	15276060
15276080	15290000	15302000	15310000	15344100	15344200	15351200	15352000	15353000	16121000	16124000
16200000	16204000	16321000	16329000	16342000	16344000	16370000	16376000	16408000	16412000	16435000

ADDRESS

0003 *04022000*

04023000 04024000

ADDRESS

0003 *04017000*

04018000 04019000

ADDRESS

0003 *04030000*

04032000 04033000

ADDRESS

0003 *05199000*

05201000

ADDRESS

0055 *07393000*

07397000 07397400 07397500 07413000

ADDRSF

0003 *01604000*

07397000 07677000 *13205000**13344000**13346000**13347000* 13348000 13373000 *13767000* 13767000 13768000
13770000 14313000 14320000 *16000500*

ADDVALUE

0003 *01000860*

02193000 *02592000**09028920*

ADES

0003 *01299010*

ADJUST

0003 *04083000*

04087000 04144000 04172000 06208500 07080000 07459000 07579000 07585100 07586000 07600200 07607100
07727040 08190000 13605000 13611000 13630000 16023100

ADJUST

0103 *16023100*

16137000 16158000

ADQP

0003 *01278000*

05344550 05344670 06013000 07666000 08029000 08031000 12013000 12022000 16005000 16326000

ADR

0091 *13633000*

13633000 13642000 13657000

ADRES

0037 *05215000*

05217000 05219000 *05219000* 05226000 05227000

AEXP

0003 *03001000*

03006000 *06002000* 06007000 06057000 06105000 06174000 06202000 06305000 06410000 06416000 06422000

07045000	07062000	07086000	07487000	07493000	08101000	08123000	08131000	08135000	08142000	12046000
12051000	12054000	13759000	15099000	15233012	15276070	15277000	15307000			

AGAIN

0016 *02360000*

02372000	02377000	02390000	02393000	02395000	02397000	02404000	02418000	02421100	02431000	02460000
02470000	02472000	02474000	02480000	02493000	02496000	02501000	02506000	02508000	02513000	02516000
02520000	02522000	02564000	02567000	02601000						

AGAIN

0044 *05439000*

05445000 05454000 05457000

AGAIN

0064 *07726000*

07727000 07727100 07741000

AGIN

0003 *02001860*

02001880 02001886

AJUMP

0003 *01596000*

13603000 *13614000* 13626000 *13629000* 14054000 *14055000**14606000*

AJUMPO

0094 *14034000*

14054000 14606000

AKKUM

0003 *01693000*

ALFAARRAYID

0003 *01204000*

ALFAID

0003 *01200000*

14139000

ALFAPROCID
0003 *01196000*

ALFASTRPROCID
0003 *01192000*

ALFAV
0003 *01284000*

ALL
0065 *08052000*
08054000 08055000 08056000 08057000

ALLTHU
0028 *04168000*
04176000 *04186000*

ALONG
0017 *02325000*
02336000 02340000 02354000 *02358000*

ALPHADEC
0094 *14013000*
14018000 *14139000*

ALPHASIZE
0003 *01707000*

ALPHAWORDS
0012 *02264700*
02264900 02265600 02265850

AMPERSAND

0003 *01266500*

06034000 06180000 07084000 15110200 15349000

ANDOP

0003 *01276000*

02964500 02971000 02983500

ANOTHER

0053 *07007000*

07009000 07012000 07015000 07024000

ANOTHER

0054 *07037000*

07041000 07095000

ARGH

0021 *02638000*

02649000 02685000 02718000 02753000 02899000

ARITHCOMP

0003 *03004000*

06036000 *06041000* 06052000 06055000

ARITHSEC

0003 *03002000*

06006000 *06011000* 06023000

ARRAE

0003 *13376000*

13410000 14156000

ARRAYDEC

0094 *14015000*

14020000 *14156000*

ARRAYMONFILE

0003 *01526000*

ARRAYV

0003 *01294000*

14161000

ASKIP

0083 *10236000*

10237000 10239000

ASSIGNDP

0003 *01273000*

06199000 07397210 07401000 07662000 08178000 08229000 12028000 13758000 15090000 15113000 15233006
15233026 15276050 15276160 15300000 15342000 16316000 16367000

ASTGG

0003 *01300100*

05348000 07600410 *07600430*

ASTRISK

0003 *01212200*

01299100 07600400 13400000 13404000 15253000

ATSIGN

0021 *02637000*

02641000 *02683000*

ATYPE

0003 *01452000*

01452000 06005000 06057000

AWAY

0008 *02191000*

02191250 02194250 02195250 *02196250*

AXNUM

0003 *07034000*

07086700 *07086700* 09384000 09384100

B

0003 *02054000*

02055000

B

0003 *01717000*

B

0003 *02062500*

02063500

B

0003 *04247000*

04248000

B

0003 *02980000*

02983500 *02983500* 02984000 02984500

B

0003 *03047100*

B

0003 *04311000*

04311070 04311080 04317000 *04317000*

B

0003 *04500000*

04526100 04528000 04542000 04544000 04548000 04551000 04556000

B

0003 *02955500*

B

0003 *03041000*

B
0003 *03050000*

B
0017 *02323000*
02343000 02352000 02353000 *02356000* 02357000

B
0024 *02973500*
*02976000**02978000**02978500* 02978500 02979000

B
0023 *02970000*
02970500 02971000 02971500

B
0033 *04505000*
04507000

B
0065 *08027000*
*08035000**08037000*

B
0065 *08010000*
08044000 08047000 08089000 08112000 08116000 *08122000**08169000* 08183000 08199000 *08209000* 08213000

B
0077 *09393240*
09393300 09393310 09393330 09393350 09393370 *09393370* 09393400 *09393410* 09393410 09393430 *09393440*
09393440

BACK
0058 *07491000*
07493000 07495000

BACK

0066 *08084000*

08089000

BACK

0082 *10259000*

10263300 10276500 10278000 10279000

BACKFIX

0066 *08091000*

*08097000**08099000**08106000* 08127000 *08146000**08147000* 08157000 08168000 08171000

BAE

0003 *03046000*

*07458000**07459000* 07459000

BANA

0003 *03048000*

05225000 *06419000* 06423000 07523000

BASE

0076 *09370000*

09371000 09374000

BASENUM

0003 *01000860*

02192500 *02581000**09028920*

BATMAN

0003 *01001600*

02961500 02966500 *02967000* 02968000 02978000

BBC

0003 *01634000*

07487000 08221000

BBW

0003 *01635000*

06207000 07444000 07495000 08047000 08078000 08166000 08168000

BEGINCTR

0003 *01499000*

07008000 *07008000**07021000* 07021000 07023000 *07024000**14035000* 14035000 *14510000* 14510000 *14520000*
14520000

BEGINV

0003 *01234000*

05108000 09252000 13771000 14393000 14479000

BETA1

0090 *13379000*

13390000 13409000

BEXP

0003 *03006000*

06410000 07487000 07493000 08135000 08142000

BFC

0003 *01636000*

06301000 07496000 07572000 07573000 07575000 07579000 07583000 07585100 08151000

BFW

0003 *01637000*

04218000 06307000 07088000 07512000 07516400 07518000 07526000 07529000 07572000 07586000 08125000
08127000 08146000 08148000 08215000 08228000 13263000 13274000 13606000 14041000 14451000

BIT

0003 *02307000*

02311000

BIT

0103 *16015000*

16216000

BITEDUST

0035 *05067000*

05087000 05088000 05089000 05090000

BITOP
0003 *01269000*
06087000

BLANKET
0003 *01737350*
01737500 02265100 02265250 02265600 02265700 02265850 02265950 05325450 09034500

BLKAD
0094 *14026000*
*14037000**14044000* 14600000

BLUCK
0003 *03067000*
07750000 07768200 *14001000* 14482000 14613000 16495210

BLUCKCTR
0003 *01430000*

BNS
0103 *16011000*
16122000 16124000

BNSFIX
0107 *16118000*
16121000 16141000

BOUARRAYID
0003 *01202000*

BOUID
0003 *01198000*
02967000 02977500 07047000 08057000 08058000 12042000 14140000

BOULCOMP
0003 *02955500*
02971000 *02980000* 02983000 02985000

BOULEANDEC

0094 *14013000*

14018000 *14140000*

BOULEXP

0003 *02065600*

02343000 *02969000**02971500* 02972000 02976000

BOULPRIM

0003 *02955000*

02970500 *02972500**02979000* 02979500 02982500

BOUPROCID

0003 *01194000*

14294000

BOUSTRPROCID

0003 *01190000*

14285000

BOOV

0003 *01282000*

BOTTOM

0044 *05439000*

05446000 05451000 *05455000*

BRANCH

0003 *04116000*

04120400 04120400 04122000

BRANCH

0003 *03037000*

BRANCHTYPE

0003 *03025000*

BRANCHTYPE

0003 *07537000*

07540000 07544000

BRET

0065 *08011000*

08016000 *08033000* 08033000 08034000 08035000 08046000 08094000 *08124000* *08180000* *08223000* *08232000*

BRNCH

0066 *08092000*

08148000 *08152000*

BSD

0103 *16021000*

16417000

BSS

0103 *16020000*

16415000

BTYPE

0003 *01452000*

BUG

0004 *01802000*

04279000 04282500

BUILDLINE

0002 *00504700*

02526000 02526000

BUMPL

0003 *01477000*

04216000 06207000 06297000 06300000 07444000 07459000 07487000 07493000 07495000 07540000 07577000

07585000 08047000 08078000 08106000 08146000 08168000 08190000 08221000 13628000 14039000

BUF

0003 *01695000*

13333000 13333000 14073000 *14079000**14083000* 14084000 14088000 *14088000* 14088000 14116000 14118000
14118000 14118000 *14364000**14369000*

B2D

0003 *01717000*

04149000 *04247000**04248000* 04279000 04281500 04282000 04282500 04285000 04285500 05376100 09434000
09447080

C

0003 *01562000*

02682000 02686000 *02701000* 02705000 *02758000* 02789000 *02789000**02801000**02803000* 02803000 02811000
02811000 02827000 *02842000* 02842000 02849000 02850000 *02923000* 06087000 06090000 06093000 06099100
06099200 06168000 07600300 07604120 07667000 *07667000* 07669000 07727050 08035000 16212500

C

0003 *03047100*

C

0003 *04203000*

04205000 04209000 04211000 *04216000* 04218000

C

0046 *06064000*

06064000 06066000 06194000 07035000 07036000

C

0041 *05334100*

05334100 05334500 05400000 05411100

C

0065 *08020000*

08021000

C

0092 *13673100*

13673100 13673350

CALL

0003 *01721000*

05402100 05405000 05407000 05409000 05428000 05430000 07397400 07397500 08072000 14483400

CALL

0065 *08072000*

08079000 08098000 08108000 08130000 08217000

CALLA

0003 *01720000*

01721000 05402100 05405000 05407000 05409000 05428000 05430000 07397400 07397500 14483400

CALLINFO

0003 *01722000*

07397300 *14481100* 14483300 14483400

CALLSTATEMENT

0094 *14017000*

14132000 14137300 *14505000*

CALLSWITCH

0003 *05315000*

05316000 13263000

CALLX

0003 *01722000*

07397300 *07397500* 07397500 14481100 *14481100* 14483500

CALL1?

0003 *01721000*

05402100 05405000 05407000 05409000 05428000 05430000 07397400 07397500 14483400

CARD

0003 *01557000*

02213750 02215500 02216250 09281500

CARD

0003 *02188000*

02188000 02188500 02202750

CARDCOUNT

0002 *00504150*

02234500 *02234500*

CARDLAST

0009 *02210000*

02210750 *02216000*

CARDNUMBER

0002 *00504100*

02228000

CARDONLY

0009 *02210000*

02210750 *02215250*

CARDOPTION

0016 *02363000*

02429000 02436000

CBUFF

0003 *01561056*

02213750 02214000 02215500 02215750 02216250 02216500 07025010 07029000

CD

0063 *07653500*

07654500 07655000 08083000 08084000

CDC

0003 *01638000*

15392000 15356000

CELL

0040 *05325040*

05325050 05325130 05325150 05325180 05325220 05325330 05325350 05325360 05325380 05325390 05325400

05325410 05325420

CHANGESEQ

0003 *01741200*

01741300 02193250

CHAR

0003 *01299040*

14461100

CHAR

0082 *10234000*

10242000 10252000 10255000

CHARCOUNT

0082 *10230000*

10242000 10250000 10251000 10256000 *10256000**10261000* 10285000

CHECKBIT

0003 *01000920*

01001210 02496000

CHECKER

0003 *05113000*

05120000 05216000 07070500 07398000 08054000 12044000 15085000

CHECKTOG

0003 *01001210*

CHKS08

0003 *13209000*

14289000

CHS

0003 *01639000*

06021000 08189000 08219000

CIV

0003 *01238000*

CLASS

0003 *01148000*

02705100	*02850000*	*02851000*	02886000	02920000	02924000	*02966500*	02968000	04509000	05350120	07044000
08057000	12120000	12125000	13201000	13226000	13677200	14062000	14091000	14425000	14519000	15086000
15102000	15128100	15233009	15276020	15276090	15295100	15303000	15333000	15344000	15351000	

CLCR

0003 *01330000*

02216500 *02217750* 02218000 02218250 02224500

CLCR

0009 *02202750*

02203500 02209000

CNT

0003 *02001838*

02001838 02001844

CNT

0003 *02001858*

02001858 02001866 02001870 02001892

CDC

0003 *01640000*

15095000 15290000 15344000 15355000

COCT

0063 *07653500*

07656500 07657000 07671000

CODE

0003 *01556900*

04147000	04296000	05312000	07662000	07668500	07669000	07671500	08999200	08999575	08999700	09393350
09393400	09393430	09400000	09401000	13651000	13746000	16495100				

CODE

0003 *04130000*

04130000 04134000 04231000

COUISK

0003 *01561300*

08999150 08999175 08999275 08999300 08999350 08999375 08999550 08999575 08999600 08999700 08999725
09398000

COL

0041 *05344000*

COLON

0003 *01260000*

02637000 05276000 06318000 06320000 07597500 07600100 07600600 07604020 07727010 07727030 07727070
13753100 13756000 13760000 16160000

COLON

0021 *02637000*

02641000 *02663000*

COM

0003 *01641000*

COMCOUNT

0007 *02089500*

02090500 02111500 02118000

COMMA

0003 *01261000*

06114000 07067000 07095000 07604140 07673000 08145000 08150000 08153000 10263700 10282000 12010000
12020000 12033000 12052000 12056000 12125000 13351000 13403000 13409000 13756000 13762000 14197000
14223000 14259110 14268000 14356000 14404000 15287000 15293000

COMMANTS

0007 *02092000*

02097500 02105000

COMMENTS

0007 *02092000*

02106500 02107500 02108000

COMMENTV

0003 *01253000*

02886000 02920000 08034000

COMMON

0054 *07037000*

07045000 07054000 07058000 07064000 07074000 07076000 07084000 *07094000*

COMPAR

0009 *02210500*

02216750 02218500 *02224250*

COMPARE

0003 *02188000*

02190000 02190250 02203500 02226000 02228150 02232000 02233500 02411000 02486000

COMPARECODE

0109 *16192000*

16208000

COMPLETE

0021 *02637000*

02658000 02667000 02680000 02705200 02715000 02757000 02852000 02877000 02877020 02893000 02894000
02909000

COMPOST

0021 *02639000*

02880000 *02882000*

COMPOUNDTAIL

0003 *07006000*

07033000 07768140 07768200 13772000 14412000 14512000 14524000 16489000

CONSTANA

0065 *08013000*

08015000 08074000 08099000 08181000 08188000 *08211000**08224000**08227000*

CONSTANB

0065 *08013000*

08043000 08112000 08116000 *08121000* 08183000 08194000 *08205000* 08213000

CONSTANC

0065 *08013000*

08185000 08195000 *08204000* 08218000

CONSTANTCLEAN

0003 *03034000*

04163000 04217000 07459000 07496000 07519000 07585000 08170000 08190000 13744000 14040000 14168500
14604000

CONV

0003 *02041000*

02043000 02228000 02251000 02255000 02259000 02502000 02581000 02592000

CONVERT

0003 *02248000*

02260000 02758000 02796000 02803000 02823000

COP

0003 *01371000*

02550000 04280000 04283500

CORADR

0003 *01344000*

05248500 05250000 *05250000* 09282600 09388000 13653000

CORE

0076 *09370000*

09371000 09375000 09393240

CORNER

0003 *02011000*

02011000 02012000

COUNT

0003 *01313000*

02128500 02129000 02251000 02252000 02278000 *02327000* 02366200 *02371000* 02373000 *02499000* *02590000*
02602700 *02647000* 02685000 02686000 *02691000* 02695000 *02698000* 02698000 02699000 *02699000* 02703000
02705000 02790000 02796000 02798000 02803000 02810000 02817000 02880000 02881000 *02889000* *02958500*
05325070 06062000 07668000 07670000 07671000 10241000 12109000 *12121000* 13302000 14259010 16392000

COUNT

0003 *02062500*

02062500 02064000

COUNT

0007 *02090000*

02121500

COUNT

0034 *05016000*

05017000 05028000

COUNT

0035 *05067000*

05067000 05071000 05077000 05080000

COUNT

0046 *06062000*

06074000 06075000 *06075000* 06103000 06112000 *06112000* 06119000 *06119000* 06120000 06121000 06122000

COUNT

0040 *05325070*

05325150 05325160 05325210 05325320 05325330

COUNT

0083 *10241000*

10242000 10245000 *10247000* 10247000 10249000 10252000 10255000 10256000

COUNT

0083 *10236000*

10237000 10240000

COUNTY

0007 *02089500*

02090500 02121000 02123500

CPLUS1

0003 *01713000*

CPLUS2

0003 *01714000*

CREL

0028 *04167000*

*04173000**04183000* 04187000

CRF

0103 *16010000*

16121000 16200000 16370000 16408000

CROSSHATCH

0003 *01262000*

02640000 10279000

CROSSHATCH

0021 *02640000*

02642000 *02714000*

D

0003 *02001858*

02001890 02001895 02001896

D

0003 *02001838*

02001842 02001848 02001850

D

0003 *03047100*

D
0012 *02264400*
02264500

D
0012 *02264700*
02264800

D
0028 *04166000*
04175000 04184000

D
0063 *07658500*
07659500 07660000 08020000 08027000 08052000 08063000 08072000 08080000 08083000 08084000

D
0082 *10257100*
10257200 12101000

D
0085 *12103000*
12104000

D
0098 *14254200*
14254300

D
0105 *16067000*
16070000 16079000

D
0104 *16031000*
16032000 16032000

DA

0003 *01559020*

DATE

0005 *01823000*

01823000 01825000

DATER

0005 *01823000*

01825000 01827000 01833000

DATIME

0003 *01820000*

01837000 02183000 02195000 02195750 02214200 02228750 02264950 02421000 05039600 05325490 07025020
13653500 13676000

DBLSTMT

0003 *03060000*

07753000 *12002000* 12059000

DBV

0003 *01243000*

16417000

DCINTYPE

0003 *01561420*

09430060

DCV

0003 *01248000*

16208000 16338000

DDES

0003 *01299000*

07677000

DEBLANK

0007 *02091500*

02094500 02100000

DEBUG

0003 *04277500*

04286000 04297000

DEBUGBIT

0003 *01000930*

01001220 02531000 02533000 02602500 04148000 04297000

DEBUGTOG

0003 *01001220*

02533000 02602500 04148000 04297000

DEBUGWORD

0003 *04130000*

04149000

DECK

0003 *01561500*

09435000 09447000 09447090

DECKBIT

0003 *01000940*

01001230 02474000 09405000 09414010 09421000 09447010 13651100

DECKTOG

0003 *01001230*

09405000 09414010 09421000 09447010 13651100

DECLARATORS

0003 *01214000*

07767000 10277000 14126000 14370000 14396000 14516000 16434000

DECLARELABEL

0003 *16000100*

16253100 16482200

DECLSW

0094 *14018000*

14134000

DEFINEARRAY

0003 *01491000*

02202030 02202040 02202050 02202060 02212500 02217250 02217500 02217750 02218000 02366100 02602600
02719000 02721000 02900000 02902000 02903000

DEFINETR

0003 *01481000*

02715000 *10262000* 10278000 *10278000* 10280000 10283000 *10283000**10284000*

DEFINEDEC

0094 *14017000*

14021000 *14254000*

DEFINEDID

0003 *01180000*

02894000 12120000 14259010

DEFINEGEN

0003 *10228000*

10286000 12124000 14266000

DEFINEINDEX

0003 *01481000*

02212500 02718000 02719000 *02721000* 02721000 02898000 02900000 02902000 02903000 *02905000* 02905000

DEFINEV

0003 *01298000*

10278000

DEL

0003 *01642000*

DEPTH

0003 *05400000*

*05402100**05403000* 05407000 *05409000* 05426000 05431000

DESC

0065 *08016000*

08098000 08108000

DEST

0003 *02045000*

02048000

DIA

0003 *01680000*

04315000 04550000

DIALA

0003 *01502000*

*04315000**08152000*

DIALB

0003 *01502000*

*04317000**08152000*

DIB

0003 *01681000*

04317000

DIQ

0003 *01237000*

DISK

0003 *01561400*

01561400 01561600 01828500 09425900 09429000 09430075 09430100

DISK

0076 *09370000*

09371000 09376000

DISKADR

0003 *01344000*

05252000 *05254000* 05254000 *09253100* 09389000 13653000

DOIT

0041 *05334100*

05350160 05350200

DOIT

0092 *13673100*

13673450 13677300

DOLLAR

0021 *02639000*

02641000 *02729000*

DOLLARCARD

0003 *02065500*

02229000 02231000 *02319000* 02604000 02730000

DOLLARTOG

0003 *01001540*

*02366000**02603000*

DOLLAR2TOG

0003 *01691500*

*02211800**02230100* 02234600 02421000

DDSTMT

0003 *07482000*

07488000 07759000

DOT

0003 *03015000*

06178000 *06338000* 06346000

DOT

0021 *02637000*

02641000 *02668000*

DOTSYNTAX

0003 *05270000*

05286000 05286000 06341000 15112000 15233005 15276150 15341000

DOV

0003 *01225000*

07494000 08159000 08186000

DPTOG

0003 *01690000*

02253000 02784000 02794000 02829000 *12012000**12016000*

DSKIP

0003 *02045000*

02046000 02049000

DSS

0103 *16362000*

16398000 16491000

DSV

0003 *01240000*

DTYPE

0003 *01452000*

DUMMY

0003 *01001470*

DUMPDEC

0094 *14014000*

14019000 *14149000*

DUMPEE

0003 *01547000*

DUMPINFO

0003 *02264000*

02266100 02468000

DUMPPDR

0003 *01552000*

DUMPPV

0003 *01287000*

DUP

0003 *01643000*

07524000 15095000 15306000

DWA

0012 *02264100*

02265500 02265600 02265700

E

0003 *03054000*

04099000 12123000 *13293000* 13333500 13342000 13755500 16000600

E

0003 *04500000*

04526200 04530000 04536000 04543000 04549000 04555000

E

0026 *04099000*

04100000 04102000 *04112000*

E

0033 *04504000*

04506000 04508000 *04509000* 04509000 04522000

E
0085 *12103000*
12103000 12105000

E
0103 *16065000*
16070000

E
0103 *16085000*
16089000 16093000

EDITLINE
0003 *02183500*
02187000 02195000 02195750 02214200 02228750 02421000 05038000 07025020

EDOC
0003 *13683000*
13685000 13687000 13693000 13715000 13716000 13732000 13733000 14002000 14003000

EDUCINDEX
0003 *01710000*

EL
0046 *06061900*
06104300 06107200

EL
0085 *12102000*
12112000 12114000 *12125000* 12126000

ELBAT
0003 *01319000*
02265100 02265250 02910000 02916000 02920000 02923000 02924000 05089000 05189000 05216000 05217000
05218000 05222000 05280000 05281000 05344570 05344615 05344620 05344660 05344700 05344710 05350120
05350160 05350180 06016000 06044000 06072000 06104300 06108000 06166000 06185000 06324000 06325000
06326000 07031000 07070000 07070500 07071000 07396000 07430000 07516000 07516300 07518000 07521000

07554000	07566000	07572000	07583000	07599000	07667000	07745000	07768100	07994000	07995000	08029000
08034000	08037000	08176000	09424000	09425900	09427500	09429000	09432000	09433000	09434000	09435000
09437000	09447000	09447010	09447070	09447080	09447090	10277000	12015000	12024000	12036000	12044000
12125000	13200000	13361000	13372000	13394000	13767000	13768000	14062000	14129000	14220000	14221000
14314000	14349000	14352000	14353000	14399000	14519000	15077000	15276010	16000700	16121000	16124000
16159000	16200000	16204000	16208000	16211000	16212500	16254000	16321000	16329000	16342000	16344000
16370000	16376000	16379000	16408000	16412000	16435000	16449000				

ELBATWORD

0003 *05113000*

05116000 05118000

ELBATWORD

0003 *05207000*

05208000 05209000

ELBATWORD

0062 *07595000*

07599000 07601000

ELBW

0059 *07503000*

07521000 07527000

ELCLASS

0003 *01328000*

05002000	*05003000*	05109000	05304000	05344550	06004000	06013000	06034000	06036000	06048000	06052000
06055000	06076000	06084000	06104100	06105000	06106300	06107100	06114000	06115000	06138000	06139000
06141000	06146000	06151000	06175000	06180000	06198000	06302000	06316000	06340000	06411000	06416000
06422000	07012000	07013000	07018000	07021000	07026000	*07032000*	07052000	07053000	07075000	07084000
07095000	07096000	07399000	07404000	07409000	07431000	07485000	07494000	07506000	07507000	07511000
07550000	07553000	*07556000*	07578000	07600300	07600400	07604110	07665000	07665500	07666000	07666500
07668000	07727000	07727010	07727030	07727050	07728000	*07749000*	*07751000*	*07768130*	08031000	08032000
08038000	08103000	08113000	08128000	08133000	08139000	08145000	08150000	08153000	08159000	08182000
08184000	08186000	08229000	10258100	12022000	12028000	12033000	12035000	12039000	12042000	12052000
12056000	13220000	13331000	13337000	*13338000*	13338000	13391000	13400000	13403000	13409000	13756000
13758000	13760000	13762000	13763000	14137000	14165000	14197000	14259020	14259120	14260000	14325000
14326000	14336000	14357000	14370000	14396000	14404000	14479000	*14516000*	15078000	15083000	15110200
15111000	15233003	15233026	15276010	15276140	15287000	15294000	15339000	15349000	16119000	16131000
16132000	16198000	16204000	16210000	16212500	16213000	16230000	16253000	16315000	16319000	16320000
16326000	16330000	16335000	16336000	16337000	16338000	16339000	16341000	16374000	16379000	16381000
16410000	16415000	16417000	16437000	16477000	16478000	16481000	16482100	16482300	16482500	

ELCLASS

0082 *10258100*

10263400 10263700 10264100 10264300 10264410 10277000 10279000 *10282000*

ELCLASS

0087 *13220000*

13226000 13246000 13278000

ELSEBRANCH

0050 *06295000*

06300000 06307000

ELSEV

0003 *01227000*

06302000 07567000 07578000 16220000 16230000

EMIT

0003 *03028000*

04004000	04007000	04014000	04019000	04024000	04086000	04221000	*04291000*	04302000	04315000	04317000
04318000	04531000	04535000	04550000	04551000	04554000	06054000	06054100	06082000	06087000	06089000
06099100	06104600	06148000	06166000	06191000	06308000	07093100	07431000	07436000	07544000	07604070
07995000	08049000	08050000	08122000	08148000	08151000	08227000	14451100	15376700	16080000	16122000
16133000	16219000	16232000	16266000	16387000	16394000	16453000				

EMITB

0003 *03036000*

04114000	04125000	04218000	06207000	06301000	06307000	07088000	07444000	07487000	07495000	07496000
07540000	07579000	07585100	07586000	07608000	08047000	08078000	08086000	08127000	08146000	08168000
08215000	08221000	13606000	14041000							

EMITC

0003 *04010000*

04014000	14451000	16032000	16079000	16121000	16124000	16200000	16208000	16211000	16212500	16213000
16216000	16342000	16352000	16370000	16378500	16379000	16383000	16408000	16415000	16417000	

EMITD

0003 *03050000*

04311000	04322000	04556000	05203000	06332000	07093000	07093200	15100000	15233013	15276070	15308000
------------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

EMITI

0003 *04500000*

06344000	15124000	15233018	15276210	15370000
----------	----------	----------	----------	----------

EMITJUMP
0103 *16065000*

16093000 16255000

EMITL
0003 *04003000*

04032000 04205000 04544000 05222000 06110000 06206000 08046000 08067000 08077000 13264000 13267000
13773000 14506500

EMITLNG
0003 *04098000*

04112000 06148000 07573000 07575000 08150000 08220000

EMITN
0003 *04022000*

04024000 04212000 04524000 05202000 07092000 08075000 08179000 08212000 12036000 15094000 15110200
15233011 15276080 15302000 15344100 15351200 15353000 16261000

EMITNUM
0003 *03049000*

04121000 *04203000* 04225000 06168000 07089000 07442000 08021000 08089000 12014000 12015000 13229000

EMITO
0003 *04006000*

04018000 04023000 04033000 04034000 04109000 04112000 04122000 04212100 04223100 04311090 04507000
04531000 04537000 04545000 05226000 05316000 06021000 06077000 06110300 06172500 06187000 06203000
07078000 07087000 07091000 07408000 07434000 07512000 07516400 07524000 07525000 07526000 08044000
08049000 08050000 08065000 08068000 08070000 08079000 08094000 08123000 08125000 08126000 08166000
08189000 08214000 08219000 08228000 12024000 12038000 12054000 13263000 13264000 13268000 13740000
13743000 13752000 13761000 13765000 13766100 14167000 14389000 14451000 15095000 15106100 15106200
15276090 15290000 15302000 15303000 15306000 15309000 15310000 15333000 15344000 15344100 15351200
15351300 15354000

EMITPAIR
0003 *04028000*

04034000 04523000 05201000 05227000 07071000 07090000 08090000 13214000 13230000 15107000 15233010
15233014 15267000

EMITSTORE
0003 *13214000*

13230000

EMITV

0003 *04017000*

04019000	04184000	04184500	04521000	05219000	05226000	05316000	07413000	07439000	07516300	08021000
08044000	08075000	08166000	08212000	12047000	13236000	13739000	15097000	15110300	15276060	15290000
15344200	15352000									

EMITWORD

0003 *04142000*

04152000 04187000 07604120

EMIT21

0033 *04503000*

04526200 04530000 04536000 04543000 04549000 04555000

ENDOFITALL

0003 *05101100*

16495210

ENDREADTAPE

0010 *02201510*

02202010 *02202080*

ENDTOG

0003 *01415000*

02653000 02653000 02680000 02884000 *06021000* 06022000 *06022000**07016000* 07019000 *07020000*

ENDV

0003 *01228000*

05109000 07013000 07018000 07748000 07749000 16477000

ENS

0103 *16013000*

16133000

ENTER

0003 *03069000*

13390000 *13714000* 14138000 14139000 14140000 14141000 14162000

ENTRY

0003 *03055000*

13314000 13729000 14165000 14194000 14207000 14259010 14319000 14333000 14403000 14409000

EOF

0009 *02210250*

02216250 *02217000*

EOF

0067 *08999050*

08999150 08999375 *08999400*

EOF

0077 *09394000*

09399000 *09413000*

EOFT

0010 *02201510*

02201750 *02202020*

EPART

0021 *02639000*

02810000

EQL

0003 *01644000*

EQUAL

0003 *02062500*

02064000 02064500 02881000

EQVDP

0003 *01274000*

02965500 02971000 06036000

ERR

0003 *01718000*

02211500 04145000 04300000 *05105000* 05109000 05110000 05285000 06005000 06086000 06095000 06107000

06109000	06116000	06142000	06151000	06154000	06176000	06302000	06334000	06411000	06415000	06417000
06421000	06423000	07015000	07053000	07097000	07099000	07405000	07412000	07437000	07485000	07494000
07513000	07517000	07598000	07600700	07604130	07667500	07672500	07674000	07727080	07732000	07994000
08039000	08134000	08160000	08178000	08229000	12007000	12033000	12039000	12052000	12056000	14259090
14259120	15081000	15115000	15119000	15233008	15256000	15261000	15276170	15276190	15294000	15297000
15343000	15364000	15367000	16125000	16132000	16160000	16205000	16206000	16207000	16214000	16217000
16253100	16316000	16347000	16367000	16384500	16397000	16418000	16483000			

ERRMAX

0003 *01001550*

02211500 *02502000* *09028910*

ERRNUM

0003 *01716000*

ERRNUM

0003 *01718000*

ERRNUM

0003 *05105000*

05106000 05107100 05107200

ERRNUM

0003 *05012000*

05044000 05091000

ERRNUM

0034 *05016000*

05017000 05026000

ERROR

0007 *02091500*

02114000 02119000

ERROR

0003 *05110000*

07667500 07672500 07674000

ERROR

0022 *02958000*

02959000

ERRORCOUNT

0002 *00501000*

02211500 *05035000* 05035000 05087000 09028000 *09028900* 09386000 09391000

ERRORTOG

0003 *01412000*

05032000 *05047000**05119000**07009000**07063000**07645000**07750000**13282000**14130000**14443000* 14511000
16130000

EXAMIN

0002 *00511000*

00512000 00529000 00531000 00533000 00536000 02214200 02225750 02228250 02230000 02230100 02665000
02680000 02697000 02745000 02750000 02753000 02786000 02791000 02807000 02813000 02888000

EXIT

0003 *01758000*

01763000 01764500 01782000

EXIT

0009 *02210000*

02215750 *02225000*

EXIT

0007 *02091500*

02104500 02112000 02114500 02115500 02118500 02119500

EXIT

0016 *02360000*

02375000 02411000 02413000 02486000 02488000 *02602000*

EXIT

0022 *02957000*

02958000 02959000 02962500 *02967500*

EXIT

0035 *05049000*

05094000 *05100000*

EXIT

0038 *05273000*

05284000 *05286000*

EXIT

0032 *04311010*

04311100 *04322100*

EXIT

0033 *04501000*

04526200 04532000 04538000 04546000 04552000 *04558000*

EXIT

0044 *05439000*

05454000 05457000 *05458000*

EXIT

0041 *05334000*

05344760 05372000 *05375000*

EXIT

0049 *06195000*

06200000 06204000 *06209000*

EXIT

0046 *06061000*

06069000 06086000 06096000 06107000 06109000 06116000 *06125000*

EXIT

0047 *06137000*

06142000 06149000 06154000 06176000 *06182000*

EXIT

0040 *05325030*

EXIT

0054 *07037000*

07097000 07099000 *07101000*

EXIT

0059 *07504000*

07514000 07516600 07517000 07519000 *07531000*

EXIT

0051 *06315000*

06333000 *06335000*

EXIT

0052 *06339000*

06341000 *06346000*

EXIT

0055 *07394000*

07402000 07405000 *07425000*

EXIT

0060 *07549000*

07555000 *07557000*

EXIT

0062 *07594000*

07598000 07600700 *07646000*

EXIT

0061 *07562000*

07573000 07574000 07576000 07579100 07584000 *07587000*

EXIT

0064 *07726000*

07727080 07732000 07733000 07735000 07737000 07739000 07743000 07746000 07749000 07751000 07753000
07755000 07757000 07759000 07761000 07763000 07765000 07768160 07771000

EXIT

0065 *08017000*

08092000 08178000 08222000 08229000 *08232000*

EXIT
0066 *08092000*
08134000 08160000 *08172000*

EXIT
0084 *12005000*
12033000 12040000 12052000 12056000 *12058000*

EXIT
0082 *10259000*
10263900 10275000 *10284000*

EXIT
0103 *16475000*
16484000 16487000 16490000 16491000 16492000 16493000 *16494000*

EXIT
0109 *16193000*
16205000 16206000 16207000 16214000 16217000 *16239000*

EXIT
0102 *15076000*
15081000 15110400 15112000 15115000 15119000 15233005 15233008 15233024 15256000 15261000 15275000
15276120 15276150 15276170 15276190 15276230 15294000 15297000 15305000 15337000 15341000 15343000
15365000 15367000 *15377000*

EXIT
0107 *16117000*
16125000 16132000 *16144000*

EXIT
0112 *16365000*
16367000 16384500 *16398000*

EXIT
0110 *16251000*
16253100 *16270000*

EXIT

0111 *16313000*

16316000 16342000 16347000 *16355000*

EXPECT

0003 *03009000*

EXPECT

0003 *06060000*

06119000

EXPRSS

0003 *03007000*

06057000 06299000

F

0003 *01688000*

05068000 *14077000**14078000**14115000* 14115000 16031000

F

0003 *02262000*

02262000 02263000

F

0018 *02441000*

02443000

F

0018 *02446000*

02448000	02635000	02955500	02980000	03009000	03011000	03019000	03020000	03024000	03025000	03028000
03029000	03036000	03037000	03038000	03039500	03041000	03042000	03047000	03047100	03049000	03050000
03051000	03051001	03053000	03055000	03057000	03067000	03068000	03070000	03071000	04003000	04006000
04010000	04011000	04017000	04022000	04029000	04030000	04115000	04116000			

F

0035 *05068000*

05069000 05076000

F

0067 *08999075*

08999100

F

0077 *09393700*

09393710 09393720 09393730 10228000 10234000

F

0092 *13673100*

13673100 13673350

F

0104 *16031000*

FACTOP

0003 *01299100*

15253000

FCR

0003 *01330000*

02182750 02194000 02195000 02195750 *02214000* 02214200 *02225250* 02225750 02228250 02228750 02230000
02230100 02421000 02731000 05038000 *07025010* 07025020 *07025030*

FCR

0003 *02016000*

02016000 02017000

FEIL

0003 *04130000*

04132000

FEJ

0018 *02441000*

02444000 02445000 02452000

FETCH

0003 *02262000*

02263000 08999175 09400000 09401000

FIEL
0003 *02056000*
02058000 02059000

FIEL
0076 *09364000*
09365000 09366000 09367000

FIEL
0091 *13633000*
13636000 13637000

FIEL
0092 *13661000*
13665000 13667000

FIL
0076 *09370000*
09372000

FILEDEC
0094 *14015000*
14020000 *14158000*

FILEID
0003 *01299300*
14430000

FILEV
0003 *01296000*

FILLIT
0040 *05325040*
05325440 05325460 05325470 05325480

FILLSTMT

0003 *07647000*

07677500 13396000

FINAL

0082 *10258300*

10263900 10264400 10264420

FINALAX

0076 *09378000*

09384100

FINDOPTION

0003 *02307000*

02317000 02318000 02327000 02967000

FINI

0103 *16475000*

16476000 16478000 16483000 16486000 16489000 *16495000*

FINIS

0007 *02091500*

02095500 02096500 02099500 02110000 02125500

FINISHED

0029 *04204000*

04220000 *04225000*

FINISHNUMBER

0021 *02638000*

02847000

FIRST

0003 *05271000*

05280000 05282000

FIRST

0051 *06314000*

06324000 06327000 06332000

FIRST

0052 *06339000*

06341000 06344000

FIRSTTIME

0009 *02210000*

02210750 *02213500*

FIRSTX

0003 *01621000*

13608000 *13612000* 13623000 14051000 *14052000* 14600000 *14608000*

FIRSTXD

0094 *14033000*

14051000 14608000

FIX

0018 *02446000*

02451000 02454000

FIX

0056 *07427000*

*07430000**07433000* 07439000 07440000

FIX

0066 *08082000*

08090000 08157000 08171000

FIXC

0103 *16029000*

16057000 16141000 16233000 16236000 16238000

FIXDEFINEINFO

0003 *01717900*

02896000 *12101000**12108000*

FIXHDR

0077 *09393700*
09393750 09430075

FIX1
0109 *16195000*
16218000 16233000 16238000

FIX2
0109 *16195000*
16232000 16236000

FL
0003 *01420000*
06140000 07050000 07058000 07073000 07083000 12037000 15110100 15276210

FLAG
0003 *01716000*
02226500 02316000 02421000 02501000 02593000 02599000 02650000 02686500 02696000 02824000 02828000
02899000 02958000 02959000 02962000 02976500 02977500 04013000 04176000 *05012000* 05101000 05106000
05119000 05344610 05344661 05347000 05348000 05370000 06104520 06110200 06200000 07024000 07025000
07061000 07600000 07600420 07600900 07604000 07604068 07671500 07728000 07747000 09282600 10243000
10264420 12025000 12030000 12115000 12126000 13210000 13282000 13332000 13338000 13340000 13386000
13391000 13395000 13400000 13404000 13651100 13729000 13753000 13754000 13758000 13761000 13763000
13766000 14066000 14135000 14136000 14137500 14146000 14153000 14161080 14165000 14168000 14187000
14220000 14263000 14286000 14295000 14311000 14326000 14333300 14337000 14339000 14346000 14351000
14358000 14371000 14442000 14486000 14508000 15376600 16033000 16071000 16163000 16171000 16265000
16434100

FLIPFLOP
0093 *13750500*
13752000 *13755500* 13755500

FOOT
0066 *08091000*
08162000 08166000

FORLIST
0065 *08080000*
08156000 08172000 08231000

FORMAL

0003 *01149000*

05209000 05218000 05236000 07070000 09133800 13202000 13244000 14427000

FORMALF

0003 *01591000*

07400000 07411000 *13202000**13328000**13341100**13343000* 13372000 *16000400*

FORMALNAME

0102 *15076100*

15092000 15105000

FORMALV

0065 *08014000*

08056000 08065000 08098000 08108000 08130000 08179000 08212000 08217000 08227000

FORMATBIT

0003 *01000950*

01001240 02567000 14461500 14493500

FORMATOG

0003 *01001240*

14461500 14493500

FORSTMT

0003 *03052000*

07755000 *08008000* 08232000

FORV

0003 *01223000*

FORWARDBRANCH

0066 *08091000*

08152000 08157000 08169000 08171000

FORWARDV

0003 *01254000*

14467000

FORWART

0066 *08084000*

08086000

FOULED

0003 *01583100*

04119500 04119500 *04151000*+04207500**07483500**07491500**07604010**14505000*

FOUND

0015 *02309000*

02313000 *02315000*

FOUND

0029 *04204000*

04209000 *04221000*

FP

0003 *01420000*

06164000 15119000 15349000

FPART

0021 *02639000*

02682000 *02790000*

FR

0003 *01420000*

08228000 15305000

FR

0003 *03042000*

FR

0003 *13657000*

13677100 13677250 *13677250* 13677250 13677300

FRMTID

0003 *01182000*

FROM

0003 *04116000*

04120000 04120100 04120500

FROM

0003 *03037000*

FROM

0003 *03038000*

FROM

0003 *03029000*

FROM

0003 *03011000*

FROM

0003 *07391000*

07402000 07404000

FROM

0003 *06194000*

06198000

FRONT

0058 *07491000*

07493000 07496000

FRSTLEVEL

0003 *01402000*

05116000 *14478000*

FS

0003 *01420000*

07739000 15102000 15104000 15114000 15233008 15233015 15276100 15276110 15276160 15276190 15334000
15335000 15342000 15364000 15367000

FUNCMONFILE

0003 *01543000*

FUNCTOG

0003 *01586000*

13736000 14054000 14187000 *14275000**14288000**14296000* 14411100 14415100 14452000 14478000 *14599000*

FUNCTOGO

0094 *14034000*

14054000 14599000

FWDTOG

0099 *14274000*

*14276000**14312000* 14333100 14373000

FZERO

0003 *01692000*

05344592 05344662 05375100 14078000

G

0003 *01610000*

13231000 13236000 *13718000* 13723000 *13724000* 13726000 *13727000* 13729000 *14084000* 14085000 14091000
14115000 14116000 14117000 *14278000**14281000* 14284000 14292000 14294000 *14296000* 14298000 14299000
14310000 14315000 *14349000* 14349000 *14352000* 14353000 *14446000* 14449000 *14470000* 14470000

GENERATE

0111 *16313000*

16322000 *16345000*

GEQ

0003 *01647000*

08050000

GET

0003 *03019000*

04102000 04105000 04182000 *05307000* 05312000 *05312000* 07604050 07607000 07608000 08086000 13235000
13258000 13267000 13273000 14168000 16032000 16092000

GETF

0002 *00523000*

00524000 00526000 00527000 00529000

GETLP

0095 *14076000*

14084000 14089000

GETSPACE

0003 *03051000*

05132000 *05331000**05375000* 05378000 08162000 13347000 14044000

GETSYL

0039 *05309000*

05310000 05312000

GETVOID

0003 *01756000*

01784000 02410000 02485000

GIT

0003 *01719000*

02901000 *05123000**05124000* 06104300 07410000 07412000 07438000 07441000 07527000 07539000 07601000
13249000 13257000 14388100 14438000 14483400 15376300 16070000 16089000 16138000 16163000 16254000
16257000

GNAT

0003 *03020000*

*05128000**05131000**05132000* 05133000 05316000 07516300 13267000

GNC

0007 *02091500*

02097000 02104000

GQGEN

0003 *03023000*

07518000 07529000 *07535000* 07545000 07572000 07573000 07575000 07583000 13274000

GOMCP

0059 *07504000*

GOSTMT

0003 *07501000*

07531000 07763000

GOT

0077 *09393060*

09393130 09393170 09393190

GOTOS

0103 *16249000*

16270000 16454000 16487000

GOTSCHK

0094 *14016000*

14160000

GOTSTORAGE

0003 *13733000*

GOTSTORAGE

0094 *14025000*

14160000 *14160000*

GOV

0003 *01232000*

07550000 07556000

GRINCH

0077 *09393060*

09393130 09393170 09393190

GRINCH

0077 *09393250*

09393350 09393400 09393430

GS

0003 *05325010*

05325460 05325470 05325480

GS

0041 *05344000*

05344592 *05344600* *05344615* *05344620* 05344620 *05344660* 05344661 05344662 *05344662* 05344662 *05344700*
05344720 *05344740* 05344740 05344750 *05350000* 05350160 05350200 *05371000* 05373000 *05373000* 05375000
05375100 *05376000* 05376000 05376100

GTA1

0003 *01697000*

13210000 13351000 13381000 13408500 13718000 13724000 13727000 14069000 14129000 14134000 14161000
14161020 14164000 14259000 14259010 14259015 14259080 14278000 14281000 14296000

GTI1

0003 *01384500*

02192500 *02193000* 02193250

GTR

0003 *01650000*

GT1

0003 *01384000*

02203500 02205000 *02452000* 02456000 *02652000* 02653000 02655000 *02658000* 02659000 02716000 *02719000*
02720000 02723000 *02823000* 02830000 *02832000* 02832000 02833000 02834000 02838000 02839000 *02875000*
02876000 02877010 02878000 02879000 02881000 02882000 *02886000* 02894000 *04105000* 04107000 04109000
04110000 *04120100* *04120500* 04121000 04122000 *04508000* 04521000 04523000 04524000 *05116000* 05117000
05231000 05233000 05235000 *05235000* *05447000* 05449000 *07042000* 07043000 07044000 *07066000* 07067000
07436000 *07539000* 07540000 07544000 07545000 *07600950* *07727020* 07727070 *07994000* 07995000 *08058000*
08060000 *09034100* 09034200 09133800 *09133900* 09134000 09134100 09134200 *09253000* 09253500 09254000
09255000 *09414040* 09414080 09414090 *09415000* 09415500 09416000 *09416000* 09416000 09417000 *09417000*
09417000 09418000 *10250000* 10251000 *13226000* 13232000 13240000 13244000 13248000 *13251000* *13256000*
13259000 *13273000* 13274000 *13280000* 13282000 *13371000* 13373000 *14039000* 14041000 14116000 14203000

14333300 *14425000* 14427000 14429000 14430000 14437000 14447000 *14483200* 14483300 *15102000* 15106200
15107000 *15295100* 15296000 16157100 *16254000* 16255000 16257000 16258000 16260000 16265000 16452000

GT1

0097 *14203000*

14206000 14221000 *14221000* 14224000

GT1

0108 *16157100*

16159000 16163000 16164000 16166000 16171000 16173000

GT2

0003 *01384000*

02721000 02723000 02834000 02839000 *02876000* 02877010 02878000 02879000 02881000 02882000 *05233000*
05234000 *05448000* 05450000 *07539000* 07545000 07595000 *07601000* 07604040 07604050 07604060 07604070
07604090 *07604090* 07605000 07607000 07607100 07608000 *07609000* 09133800 09133900 09134000 *09134100*
13284000 13285000 13286000 14203000 *14430000* 14431000 *14446000* 14448000 *16163000* 16167000 16172000
16254000 16255000 16266000

GT2

0097 *14203000*

14217000 14224000

GT3

0003 *01384000*

01721000 05402100 *05402100* *05405000* 05405000 *05407000* 05407000 *05409000* 05409000 05428000 *05428000*
05430000 05430000 07397400 *07397400* *07397500* 07397500 *07527000* 07528000 07595000 *07601000* 07610000
09134100 09134200 *13285000* 13287000 *14438000* 14438000 *14483400* 14483400

GT4

0003 *01384000*

02904000 *06104300* 06104500 *06104500* *06104510* 06104510 *06104520* 06104600 *07527000* 07527000 07528000
07529000 07596000 *07601000* 07603000 07610000 *13235000* 13238000 14203000

GT4

0097 *14203000*

GT5

0003 *01384000*

07028000 07029000 07596000 *07604050* 07604080 07604090 *07607000* 07609000 14203000

GT5

0097 *14203000*

H

0005 *01822000*

01828000 01833000 01835000

H

0003 *05315000*

05316000

H

0035 *05068000*

05071100 05071200 05085000

HEXIZE

0003 *02001858*

02001897 02001898

HEXOP

0003 *01268000*

01299200 06071000 06089000

HF

0094 *14017000*

14123000 *14382000* 14508000

HOLE

0055 *07393000*

07396000 07398000 07410000 07412000

HTTEOAP

0003 *13731000*

13747000 14489000 14600000

I

0003 *01319000*

02917000	02917000	04237000	*05002000*	05002000	05003000	*05003000*	*05107000*	05107000	*05107100*	05107100
05107200	05107200	05189000	05216000	05217000	05218000	05221000	05222000	05224000	*05225000*	05225000
05280000	05281000	05344570	*05344600*	05344600	05344615	05344620	05344660	*05344670*	05344670	05344700
05344710	05344770	*05344770*	05350120	05350160	05350180	05414000	05438000	06016000	06044000	06072000
06104300	06108000	06166000	06185000	06324000	06325000	06326000	*07008000*	07008000	07031000	*07031000*
07066000	07070000	07070500	07071000	07396000	07397210	07401000	07430000	07516000	07516300	07518000
07521000	07554000	07556000	*07556000*	07566000	07572000	07583000	07599000	*07600200*	07600200	*07600500*
07600500	07600950	07604020	*07604130*	07604130	*07604150*	07604150	07667000	07727040	*07727040*	*07727060*
07727060	07745000	07748000	07749000	*07749000*	*07750000*	07750000	07751000	*07751000*	07768100	*07768130*
07768130	07768170	*07768170*	*07768180*	07768180	07994000	07995000	08029000	08034000	08037000	*08110000*
08110000	08111000	*08120000*	08120000	*08174000*	08176000	*08223000*	08999050	08999500	*09391000*	09393270
09414030	09414040	09414100	*09414100*	*09415000*	09415000	09418000	09419000	*09419000*	09419000	*09420020*
09424000	09424000	09426000	*09427000*	09427100	09427500	*09427500*	*09432000*	09433000	09434000	*09436000*
09436000	09447020	09447030	09447040	10258200	12015000	12020000	12024000	12036000	12044000	13200000
13324000	*13324000*	13361000	13372000	13394000	13753100	13757000	*13757000*	13761100	13767000	13768000
14062000	14063000	*14067000*	14067000	14129000	14137200	*14137200*	14189000	*14189000*	14220000	14221000
14255000	14255000	14264000	*14264000*	14314000	14332000	*14332000*	14349000	14352000	14353000	*14360000*
14360000	14399000	14408000	*14408000*	14516000	14519000	15077000	15276010	*15276040*	15276040	16000700
16121000	16124000	16159000	16200000	16204000	16208000	16211000	16212500	*16252000*	16252000	16254000
16321000	16329000	16342000	16344000	16370000	16376000	16379000	16408000	16412000	16435000	16449000
16449000	*16450000*	16450000	16482300	*16489000*	16489000	16495100				

I

0003 *05400000*

05402000 05402200 05402300 05404000 05409200 05409500

I

0030 *04237000*

04239000 04241000 *04242000* 04243000

I

0043 *05415000*

05415000 05419000 05437000 06060000

I

0044 *05438000*

05444000 05445000 05447000 05455000 *05456000* 05456000 05458000 05459000

I

0043 *05414000*

05424000 05425000 05425100 *05425400* 05425400 05426000 *05430000* 05430500 05431000

I

0041 *05334100*

05334400 05334600

I

0067 *08999050*

08999175 08999225 08999350

I

0068 *08999500*

08999525 08999550 08999650

I

0079 *09393270*

09393300 09393310

I

0082 *10258200*

10277000 10282000

I

0092 *13673100*

13673300 13673400

ID

0015 *02310000*

02312000 02313000

ID

0035 *05067000*

05067000 05070000 05246000 05247000 05325010

ID

0040 *05325040*

05325320

IDBIT

0054 *07039000*

07047000 07050000 07068000 07082000

IDBLDR

0007 *02091500*

02095000 02103000 02110500

IDENT

0021 *02639000*

02643000 *02865000*

IDMAX

0003 *01479000*

05344520 06106300 07664000 13337000 13754000 16213000 16253000 16482100

IDV

0003 *01651000*

IFCLAUSE

0003 *03018000*

06296000 *06409000* 07563000

IFEXP

0003 *03013000*

06005000 *06294000**06299000* 06311000

IFS

0103 *16191000*

16239000 16486000

IFSB

0109 *16193000*

16194000 *16216000*

IFSC

0109 *16193000*

16194000 *16207000*

IFSTMT

0003 *03022000*

07561000 07587000 07761000

IFSW

0109 *16194000*

16197000

IFTDG

0109 *16193000*

16194000 16215000 *16217000*

IFV

0003 *01231000*

06004000

IMPFUN

0003 *03039000*

06158000 *06183000*

INCR

0003 *01153000*

02658000 05124000 05189000 05222000 06080000 06189000 13206000 13240000 13307000 *14116000* 15255000
15295100

INCRF

0003 *01607000*

13206000 13769000 14314000

INDEC

0094 *14014000*

14019000 14158000

INDEX

0003 *05005000*

05006000

INDEX

0003 *03053000*

INDEX

0003 *05008000*

05009000

INDEX

0003 *07036000*

07042000 07098000

INDEX

0062 *07595000*

07601000 07610000

INDEX

0111 *16314000*

*16320000**16328000**16334000* 16346000 16349000

INDEXS

0103 *16311000*

16355000 16490000

INFO

0003 *01007000*

02212250	02214250	02226000	02228000	02234750	02265600	02265700	02265850	02265950	02368000	02410000
02485000	02602800	02877010	02878000	02879000	02881000	02923000	04172000	04187000	04209000	04210000
04211000	04221000	04223000	05006000	05009000	05039500	05045000	05088000	05233000	05325470	05350160
05350200	05426000	05431000	05449000	05450000	07086400	08035000	09034200	09034500	09085000	09121650
09253500	09254000	09255000	09384100	09386000	09392500	10251000	10254000	10264800	12117000	13281000
13310000	13593000	13677300	14440000							

INFO

0003 *02005000*

02006000

INFO

0016 *02320200*

02320600

INFO

0077 *09392500*

09406000 09427500 09447060

INFO

0083 *10236000*

10238000

INLINE

0003 *03010000*

07746000 *13748000* 13774000

INLINETO

0003 *01695500*

13307500 *13752000**13770500*

INT

0065 *08012000*

08060000 08070000

INTARRAYID

0003 *01205000*

04522000 08057000 15233009 15333000

INTBIT

0003 *01000960*

01001250 *02577000* 02577000 *09253050* 09253050 09253100 09394100 09414010 09430050 13651100

INTEGERDEC

0094 *14013000*

14018000 *14141000*

INTID

0003 *01201000*

04509000 08038000 08061000 12035000 12042000 14141000 15086000 15102000

INTNAMEID

0003 *01205400*

INTOG

0003 *01001250*

02577000 09253050 09253100 09394100 09414010 09430050 13651100

INTPROCID

0003 *01197000*

13278000 14294000 15078000

INTRINSICADR

0003 *01299500*

09253100

INTRNSICPROCID

0003 *01188000*

INTSTRPROCID

0003 *01193000*

14284000

INTV

0003 *01285000*

INV

0003 *01290000*

INX

0003 *01652000*

04531000 07091000 15303000 15344100 15351200

IDCLASS

0003 *01233000*

IDSTMT

0003 *03012000*

07765000 *07993000*

IPART

0021 *02639000*

02643000 *02758000*

IS

0033 *04501000*

04527000 04540000 *04548000*

ISD

0003 *01653000*

15103000 15276100 15334000

ISKIP

0083 *10236000*

10237000 10238000

ISN

0003 *01654000*

15103000 15276100 15334000

ISULATE

0003 *01270000*

06092000

ITUP

0102 *15076000*

15233018 15233026

J

0003 *01618000*

04166000 05401000 05414000 05438000 07393100 08999050 08999500 09393010 12102000 *13210000* 13210000
13221000 *13324000* 13333500 *13333500* *13342000* 13342000 13351000 13381000 *13381000* *13408500* *13718000*
13718000 13724000 *13724000* 13727000 *13727000* 13750000 *14069000* 14129000 *14129000* 14132000 14134000
14135000 *14135000* 14136000 *14136000* 14137600 *14137600* *14161000* 14161000 *14161000* 14161000 14161020
14161020 14164000 *14164000* 14254100 *14278000* 14278000 *14281000* 14281000 *14296000* 14296000 *14421000*
14425000 14438000 14440000 14446000 14448000 *14449000* 14449000 15075000 16314000 16364000

J

0003 *10228000*

10263110 10264600 10264900

J
0028 *04166000*

04170000 04172000 04187000

J
0042 *05401000*

05405000 *05405000* 05406000 05407000

J
0044 *05438000*

05444000 05446000 05448000 05452000 *05453000* 05453000

J
0043 *05414000*

05428000 *05428000* 05428000 05429000 05430000

J
0055 *07393100*

07397300 07397400

J
0067 *08999050*

08999125 08999225 08999275 *08999275* 08999350

J
0068 *08999500*

08999625 08999650 08999700 *08999750* 08999750

J
0079 *09393270*

09393290 09393350 *09393360* 09393360 09393400 09393420 *09393420* 09393430 *09393450* 09393450

J
0077 *09393010*

09393270 09414022 09414024 09414026 *09414026* *09414030* 09414060 *09414060* 09414070 09414110 *09420020*
09425000 *09425000* *09428000* 09428000 09447050 09447060 09447080

J
0082 *10257100*

10257100 10257500 10257700

J

0087 *13221000*

13232000 13239000 *13257000* *13258000* 13258000 13264000 13265000 13270000 13271000

J

0085 *12102000*

12116000 *12116000* 12117000 12126000

J

0093 *13750000*

13765000 13765000

J

0098 *14254100*

14259010 14259015 *14259015* *14259080* 14259080 14259150 14266000

J

0098 *14254200*

14254200 14254500

J

0102 *15075000*

15234000 15255000 15266000 *15280000* 15280000 15290000 15296000 15302000 15344000 15350000

J

0111 *16314000*

16348000 16349000 *16349000*

J

0112 *16364000*

16387000 16389000 *16392000* 16392000 *16392000* 16393000

JAZZ

0102 *15076000*

15233012 15233026

JEDEN

0012 *02264100*

02265000 02265100 02265350 02265450 02265600 02265700 02265850 02265950

JFC

0103 *16016000*

16219000

JFW

0103 *16007000*

16079000 16232000

JNS

0103 *16023000*

16453000

JOINFD

0003 *01482000*

16127000 *16128000* 16134000 16138000 16139000 *16143000* 16440000 *16442000* 16445000 16447000 16449000

JOINT

0107 *16118000*

16127000 16143000

JRV

0103 *16009000*

16079000

JUMPCHAIN

0103 *16085000*

16097000 16139000 16173000

JUMPCHKX

0003 *03058000*

13616000 14165000 14387000

JUMPCHKX

0003 *03059000*

13596000 14506000

JUMPCR

0003 *01459000*

14497000 *14499000*

JUMPLEVEL

0003 *01485000*

16260000 16265000 *16269000**16433000**16435000* 16452000

JUMPS

0103 *16431000*

16455000 16493000

JUMPV

0003 *01242000*

JUNK

0003 *01430000*

K

0003 *01693000*

02091500 05401000 05414000 05438000 08016000 08999050 09393010 10258100 12102000 *14085000* 14254100
16364000

K

0007 *02091500*

02099000

K

0042 *05401000*

05405000 05406000

K

0043 *05414000*

05428000 05429000

K

0044 *05438000*

05444000 05452000 05454000 *05454000* 05455000 *05457000* 05457000

K

0067 *08999050*

08999200 08999275 *08999275* 08999275 08999350

K

0065 *08016000*

08033000 08034000 08035000 08094000 08180000 08223000 08232000

K

0077 *09393010*

09447030 09447060

K

0085 *12103000*

12103000 12104000

K

0082 *10258100*

10263200 10263800 10264200 *10264200**10264400* 10264400

K

0085 *12102000*

12109000 12117000 12118000 12121000

K

0098 *14254200*

14254200 14254400

K

0098 *14254100*

14259010 14259080 14259090

K

0112 *16364000*

16389000

KEY

0003 *04231000*
04239000 04241000

KLASSF

0003 *01603000*
05350100 05350200 *13201000* 13220000 *13226000* 13246000 13278000 *13333500* *13342000* 13372000 *13677200*
13677300 *16000300*

KNOT

0024 *02973500*
02974500 02978500

L

0003 *01428000*
01477000 01710000 01758000 02091500 02127500 02182750 02195000 02195750 02214200 02228750 02421000
04086000 04102000 04105000 *04105000* 04105000 *04109000* 04109000 04119000 *04120000* *04124000* 04145000
04145000 04147000 04149000 *04151000* 04151000 *04172000* 04172000 04175000 04182000 *04185000* *04186000*
04207500 04210000 *04216000* 04216000 04218000 04223000 04232000 04279000 04282500 04293000 04296000
04298000 *04298000* *04300000* 05038000 05049000 06054100 *06054100* 06104400 06104500 06104510 06148000
06148000 06206000 06207000 *06207000* *06297000* 06297000 *06300000* 06300000 06301000 06307000 *06308000*
06308000 07025020 07079000 07081000 07088000 07411000 *07411000* *07433000* 07433000 *07440000* 07440000
07443000 07444000 *07444000* 07459000 *07459000* 07483500 07485000 07487000 *07487000* 07491500 07493000
07493000 07495000 *07495000* 07496000 07540000 *07540000* 07545000 07577000 *07577000* 07579000 07583000
07583000 07585000 *07585000* 07585100 07586000 07597000 *07600300* *07600440* 07600800 07604010 07604040
07604060 07604067 07604085 *07604085* 07604090 *07604100* 07607100 07608000 07610000 07727020 *07727050*
07727070 08046000 08047000 *08047000* 08050000 *08050000* 08077000 *08078000* 08078000 *08089000* 08093000
08097000 08106000 *08106000* 08122000 08124000 08127000 08146000 *08146000* 08147000 *08148000* 08148000
08150000 08150000 08152000 08160000 *08168000* 08168000 08169000 08170000 08171000 08179000 *08190000*
08190000 08191000 08215000 08221000 *08221000* 08227000 08228000 *08231000* 09282600 09388000 *09395000*
09395000 09396000 09397000 09416000 *13232000* 13232000 13235000 *13238000* *13239000* 13258000 *13263000*
13264000 13267000 *13270000* 13273000 *13277000* 13360000 13606000 13612000 13628000 *13628000* 13746000
13752000 *13765000* 13765000 *13765000* *14039000* 14039000 14041000 14054000 *14055000* 14167000 14168000
14388100 14451200 14461100 14461200 14461300 *14461300* *14483700* 14487000 14505000 14602000 14603000
15376400 15376500 15376600 *16032000* 16032000 *16034000* 16070000 16088000 *16089000* 16090000 16092000
16094000 *16096000* 16121000 16138000 16167000 16172000 16218000 16232000 16257000 16313000 16364000
16475000

L

0003 *01758000*
01762000 01764000

L

0003 *02183500*
02183750 02186000 03047000 03047100

L
0003 *01742100*
01742110

L
0007 *02127500*
02128000 02135500

L
0003 *01719000*

L
0007 *02091500*
02101000

L
0003 *05333000*
05376100

L
0003 *03051001*

L
0003 *03020000*

L
0003 *03039500*

L
0003 *05307000*
05312000

L

0003 *05128000*
05131000 05132000

L
0003 *05411100*

L
0003 *05123000*
05124000

L
0003 *05437000*
05440000 05441000 05442000 05444000 05458000

L
0003 *08999450*
08999525

L
0030 *04232000*
04241000 *04243000*

L
0035 *05050000*
05051000 05052000

L
0035 *05059000*
05060000

L
0035 *05049000*
05092000 05094000 05098000

L
0046 *06064000*
06065000

[illegible]

LABELR
0003 *07593000*
07646000 07741000

LABELS
0103 *16156000*
16177000 16485000

LABELV
0003 *01286000*
14399000

LABLNONFILE
0003 *01535000*

LAMPER
0047 *06137000*
06164000 06166000 06168000 *06179000*

LASS
0102 *15076000*
15276060 15276160

LAST
0102 *15076000*
15302000 15342000

LASTCRDPATCH
0003 *01001500*

LASTENTRY
0003 *01376000*
04170000 *04189000* 04208000 04210000 04211000 *04214000* 04214000 14038000

LASTGT
0103 *16003000*
16089000 16266000

LASTGT
0101 *14420000*
14438000

LASTINFO
0003 *01619000*
02724000 02725000 02726000 *02726000* 05350200 *09033000* 10261000 10263100 10263110 10264800 13222000
13290000 13309000 *13311000* 13347000 13348000 13408000 13755500 14079000 14083000 14259150 14320000
14330000 14330000 *14373000* 14421000 *14463000* 16000700 16445000 *16447000*

LASTINFOT
0094 *14029000*
14330000 14373000 14463000

LASTRESULT
0082 *10258000*
10262000 10272000 *10276000*

LASTSEQROW
0003 *01570000*
02214250 02226000 02228000 02234750 02368000 02410000 02485000 02602800 05039500 05045000 07086400
09033000 09034200 09034500 09253500 09254000 09255000 09384100 09386000

LASTSEQUENCE
0003 *01569000*
02214250 02226000 02228000 02234750 02368000 02410000 02485000 02602800 05039500 05045000 07086400
09033000 09034200 09034500 09253500 09254000 09255000 09384100 09386000

LASTUSED
0003 *01391000*
02194500 *02204000**02209250* 02212000 02212250 *02212750* 02212750 02228050 02234000 *02332000**02430000*
*02438000**02458000**02722000* 02900000 *02901000**07030000**09029000**09037000* 09282000

LBC
0003 *01645000*

LBP
0094 *14023000*

LBU
0003 *01646000*

LCARD
0008 *02191000*
02191250 *02194750*

LCK
0003 *01330000*
02129500 02193250 *02202000* *02204000* *02209000* 02213000 *02214000* 02214250 02214500 *02215750* 02225250
02226000 02228150 02232000 02233500 02234750 02235750 02368000 02369000 02410000 02485000 02602800
02716000 02717000 *02719000* 02902000 *02903000* 02904000 02905000 *07025010* *07025030*

LCK
0003 *02005000*
02005000 02006000

LCK
0003 *01756000*
01756000 01767000 01774000

LCK
0016 *02320200*
02320200 02320600 02321000

LDES
0003 *01299030*
05248000 05250000 09415500 13252500

LDOT
0047 *06137000*
06144000 06156000 06158000 06160000 06162000 06170000 06172500 *06178000*

LEFTPAREN
0003 *01212000*
02975000 06069000 06415000 07409000 07437000 07768110 10264100 12907000 12114000 13753000 14137000

14259020 14326000 16003000 16125000 16482400

LENGTH1

0016 *02360000*

02364000 *02374000*

LENGTH2

0016 *02360000*

02380000

LENGTH3

0016 *02360000*

02364000 02381000

LENGTH4

0016 *02360000*

02364000 *02399000*

LENGTH5

0016 *02361000*

02364000 *02476000*

LENGTH6

0016 *02361000*

02365000 *02518000*

LENGTH7

0016 *02361000*

02365000 *02569000*

LENGTH8

0016 *02361000*

02574000

LENGTH9

0016 *02361000*

02365000 02575000

LEQ

0003 *01655000*

08050000

LEVEL

0003 *01402000*

05208000	05344400	05350100	05350120	13339000	13371000	13677100	14053000	*14053000*	14220000	14307000
14318500	*14323000*	14323000	14384100	14387000	14389000	14411300	14415300	14451000	14461100	14477000
14478000	14497000	14499000	*14500000*	14500000	14603000	*14604000*	14604000	16005000		

LEVEL

0103 *16005000*

16171000 16265000

LEVELF

0003 *01605000*

13204000 13339000 13371000 14307000

LFC

0003 *01648000*

LFTBRKET

0003 *01263000*

05274000 06139000 06316000 06421000 10264100 12114000 13391000 14259020 15233000

LFTPAREN

0103 *16003000*

16125000 16482400

LFU

0003 *01649000*

14167000

LGTFLD

0103 *16004000*

16172000

LIB

0009 *02202500*

LIN

0003 *01559010*

02181000	02181250	02182750	02194000	02195000	02195750	02214200	02226750	02228750	02421000	04149000
04150000	05038000	05039500	05039600	05044000	05046000	05096000	05097500	05099000	05325450	05325460
05325470	05325480	05325490	06122000	06123000	07025020	07086300	07086500	09984100	09384500	09386000
09387000	09388000	09389500	13653000	13653500	13675000	13676000				

LIN

0040 *05325040*

05325090 05331000 05333000

LINE

0003 *01558000*

01829000	02181000	02181250	02195000	02195750	02214200	02228750	02264950	02265150	02265300	02265450
02265650	02265750	02265900	02266000	02421000	02464000	04150000	04279000	04282500	05039600	05046000
05087000	05088000	05089000	05090000	05097500	05099000	05325490	05427000	05432000	05434000	06123000
07025020	07086500	09384500	09387000	09389500	09390000	13653500	13676000	14023100	14461500	14493500

LINE

0003 *02183500*

02184250

LINE

0034 *05016000*

05019000 05021000 05024000

LINE

0035 *05067000*

05069000 05071200 05076000 05085000 05246000 05247000

LINE

0054 *07038100*

07038200 07038300 07391000 07536000 07537000 07647000

LINE

0076 *09378000*

09379000 09380000 09393060 09393250

LINK

0003 *01154000*

02725000	04166000	05124000	05132000	05189000	05233000	05234000	05447000	05448000	06079000	06189000
06190000	07595000	09133800	09134000	13207000	13232000	13286000	13308000	13361000	13362000	13364000
14448000	14470000	15080000	16087000	16443000						

LINK

0028 *04166000*

04182000 04185000

LINK

0062 *07595000*

07601000	07604040	07604050	07604060	07604070	07604090	07605000	07607000	07607100	07608000	07609000
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

LINK

0106 *16087000*

16092000 16094000

LINKC

0003 *01156000*

01721000	02212250	02876000	05006000	05009000	05325470	05350180	05350200	05402100	05405000	05407000
05409000	05426000	05428000	05430000	05431000	05449000	05450000	07397400	07397500	09393110	09393330
09414070	09418000	09419000	09433000	09447060	10251000	10264800	12117000	13281000	13310000	13593000
13677300	14483400									

LINKF

0003 *01606000*

13207000 13768000 13769000 14310000 14314000 14315000

LINKP

0089 *13359000*

13361000 13362000 13364000

LINKR

0003 *01155000*

01721000	02212250	02875000	05006000	05009000	05325470	05350160	05350200	05402100	05405000	05407000
05409000	05426000	05428000	05430000	05431000	05449000	05450000	07397400	07397500	09393090	09393280
09414070	09418000	09419000	09433000	09447060	10251000	10254000	10264800	12117000	13281000	13302000
13303000	13310000	13593000	13677300	14483400						

LINKR

0079 *09393280*

09393350 09393400 09393430

LINKR

0078 *09393090*

09393130 09393170 09393190

LINKT

0089 *13359000*

13361000 13362000 13364000 13365000 *13366000*

LINKTOG

0003 *01399000*

04100000 *04213000**04222000**04295000* 06021000 *06022000*

LISTABIT

0003 *01000970*

01001260 02194250 02511000 02602500

LISTATOG

0003 *01001260*

02194250 02602500

LISTBIT

0003 *01000980*

01001270 02402000 02602500 *09028000*

LISTER

0003 *01001570*

02195000 02195750 02214200 02464000 *02602500* 05036000 07025020 07086100 *09028000* 13652000 13674000
14461500 14493500

LISTID

0003 *01181000*

LISTOG

0003 *01001270*

02602500 09028000

LISTPBIT

0003 *01000990*

01001280 02195000 02228750 02480000

LISTPTOG

0003 *01001280*

02195000 02228750

LITERAL

0003 *04003000*

04004000

LITNO

0003 *01210000*

02850000	05275000	05277000	05344610	06085000	06094000	06317000	06319000	06321000	07600300	07665000
07666500	07727050	07994000	13392000	15276010	16119000	16204000	16212500	16326000	16341000	16374000
16410000	16435000	16482300								

LITY

0003 *01250000*

16372000 16381000

LN

0093 *13750000*

13755500 13773000

LND

0003 *01656000*

04545000

LNG

0003 *01657000*

04112000

LO

0094 *14029000*

14304000 14496000

LOC
0101 *14420000*
14438000

LOC
0103 *16003000*
16070000 16163000 16255000

LOCAL
0003 *05206000*
05210000 *05210000* 07516000 07554000 07599000

LOCALV
0003 *01249000*
14396000

LOGBEGIN
0094 *14022000*

LOGFLD
0103 *16004000*
16138000

LOGLID
0003 *01179000*
13755000 14409000 14411200 14415200 14425000 16198000 16317000 16326000 16330000 16335000 16340000
16368000 16406000 16482100 16482300 16482500

LOCV
0003 *01247000*
16336000 16339000

LOD
0003 *01658000*
04523000 05201000 05227000 05316000 06172500 07071000 07090000 08079000 15233010 15267000 15306000
15344100 15351300

LOLD

0094 *14030000*

14054000

LONGID

0102 *15076100*

15303000 15344000 15351000

LOR

0003 *01659000*

LOWER

0003 *05299000*

05304000

LPRT

0003 *01483000*

LQV

0003 *01660000*

LS

0093 *13750000*

13752000 13765000

LSQ

0076 *09364000*

09366000

LSS

0003 *01661000*

LSTRTN

0003 *01430000*

LSTSEQ

0034 *05016000*

05020000 05023000

LTAPE

0008 *02191000*

02191250 *02195500*

LVL

0003 *01151000*

05116000 05208000 07432000 13204000 14220000 16005000 16171000 16265000

L1

0035 *05049000*

05096000 05099100

L1

0041 *05334000*

L1

0047 *06128000*

06133000 *06152000*

L1

0063 *07658000*

07667500 07672500 07673000 *07675000*

L1

0064 *07713000*

07720000 *07729000*

L1

0069 *09025000*

L1

0084 *12005000*
12009000 12037000

L1
0091 *13635000*
13639000 13641000

L1
0093 *13751000*
13754000 13761100 13762000

L1
0102 *15076000*
15092000 15116000

L1
0103 *16475000*
16482400 *16484000*

L1
0112 *16365000*
16372000 *16397000*

L1
0111 *16313000*
16349000 16353000

L10
0047 *06128000*
06133000 06152000

L10
0064 *07713000*
07720000 *07734000*

L10
0103 *16475000*
16476000 *16493000*

L11
0047 *06129000*
06134000 *06157000*

L11
0064 *07714000*
07721000 07729000

L12
0047 *06129000*
06134000 06152000

L12
0064 *07714000*
07721000 *07736000*

L13
0047 *06129000*
06134000 06152000

L13
0064 *07714000*
07721000 07729000

L14
0047 *06129000*
06134000 *06159000*

L14
0064 *07714000*
07721000 07729000

L15
0047 *06129000*
06134000 06159000

L15

0064 *07714000*
07721000 07729000

L16
0047 *06129000*
06134000 06152000

L16
0064 *07714000*
07721000 07729000

L17
0047 *06129000*
06134000 06152000

L17
0064 *07714000*
07721000 07729000

L18
0047 *06129000*
06134000 *06161000*

L18
0064 *07714000*
07721000 07734000

L19
0047 *06129000*
06134000 06161000

L19
0064 *07714000*
07721000 07734000

L2
0041 *05334000*
05373000

L2
0047 *06128000*
06133000 06152000

L2
0064 *07713000*
07720000 07729000

L2
0084 *12005000*
12018000 12048000 *12055000*

L2
0093 *13751000*
13753100 13754000 *13760000*

L2
0103 *16475000*
16481000 16482200 *16485000*

L20
0047 *06129000*
06134000 06152000

L20
0064 *07714000*
07721000 07729000

L21
0047 *06130000*
06135000 06152000

L21
0064 *07715000*
07722000 07729000

L22

0047 *06130000*
06135000 *06163000*

L22
0064 *07715000*
07722000 *07738000*

L23
0047 *06130000*
06135000 06163000

L23
0064 *07715000*
07722000 07738000

L24
0047 *06130000*
06135000 06152000

L24
0064 *07715000*
07722000 07729000

L25
0047 *06130000*
06135000 06152000

L25
0064 *07715000*
07722000 07729000

L26
0047 *06130000*
06135000 06163000

L26
0064 *07715000*
07722000 07738000

L27
0047 *06130000*
06135000 06163000

L27
0064 *07715000*
07722000 07738000

L28
0047 *06130000*
06135000 06152000

L28
0064 *07715000*
07722000 07729000

L29
0047 *06130000*
06135000 06152000

L29
0064 *07715000*
07722000 07729000

L3
0047 *06128000*
06133000 06152000

L3
0064 *07713000*
07720000 07729000

L3
0084 *12005000*
12026000 *12056000*

L3

0093 *13751000*
13767000 13770000

L3
0103 *16475000*
16476000 *16486000*

L30
0047 *06130000*
06135000 06163000

L30
0064 *07715000*
07722000 07738000

L31
0047 *06131000*
06136000 06163000

L31
0064 *07716000*
07723000 07738000

L32
0047 *06131000*
06136000 06153000

L32
0064 *07716000*
07723000 *07740000*

L33
0047 *06131000*
06136000 *06165000*

L33
0064 *07716000*
07723000 07730000

L34
0047 *06131000*
06136000 *06167000*

L34
0064 *07716000*
07723000 07730000

L35
0047 *06131000*
06136000 06165000

L35
0064 *07716000*
07723000 07730000

L36
0047 *06131000*
06136000 06167000

L36
0064 *07716000*
07723000 07730000

L37
0047 *06131000*
06136000 *06173000*

L37
0064 *07716000*
07723000 07730000

L38
0047 *06131000*
06136000 *06169000*

L38

0064 *07716000*
07723000 *07742000*

L39
0047 *06131000*
06136000 *06171000*

L39
0064 *07716000*
07723000 07730000

L4
0047 *06128000*
06133000 06152000

L4
0064 *07713000*
07720000 07729000

L4
0103 *16475000*
16476000 *16487000*

L40
0064 *07716000*
07723000 *07744000*

L41
0064 *07717000*
07724000 07770000

L42
0064 *07717000*
07724000 *07752000*

L43
0064 *07717000*
07724000 *07754000*

L44

0064 *07717000*

07724000 *07756000*

L45

0064 *07717000*

07724000 *07758000*

L46

0064 *07717000*

07724000 *07769000*

L47

0064 *07717000*

07724000 07769000

L48

0064 *07717000*

07724000 07769000

L49

0064 *07717000*

07724000 07770000

L5

0047 *06128000*

06133000 06152000

L5

0064 *07713000*

07720000 07729000

L5

0103 *16475000*

16476000 *16488000*

L50

0064 *07717000*
07724000 07769000

L51
0064 *07718000*
07725000 *07760000*

L52
0064 *07718000*
07725000 *07762000*

L53
0064 *07718000*
07725000 *07764000*

L54
0064 *07718000*
07725000 *07766000*

L6
0047 *06128000*
06133000 06152000

L6
0064 *07713000*
07720000 07729000

L6
0103 *16475000*
16476000 16489000

L7
0047 *06128000*
06133000 *06155000*

L7
0064 *07713000*
07720000 *07732000*

L7

0103 *16475000*

16476000 16482500 *16490000*

L8

0047 *06128000*

06133000 06152000

L8

0064 *07713000*

07720000 07732000

L8

0103 *16475000*

16476000 *16491000*

L9

0047 *06128000*

06133000 06152000

L9

0064 *07713000*

07720000 07729000

L9

0103 *16475000*

16476000 *16492000*

M

0044 *05438000*

05441000 05442000 05443000 05444000 05445000

M

0042 *05401000*

05404000 *05407000* 05407000 *05409000* 05409000 05409300 05409400

M

0041 *05343000*
05350100 05350140

M
0077 *09393010*
09447040 09447050

MACRO
0003 *01487000*
09215000 16346000 16349000

MACRO
0003 *10228000*
10263600 10276500

MACROID
0003 *01717800*
02913000 *12107000**12127000* 13305200 13307500 *14265900**14266100*

MAKEUPACCUM
0003 *01627000*
13333500 13342000 *13368000* 16000600

MARK
0003 *01615000*
14117000 *14314000**14319000* 14333300 14334000 14374000 14430000 14451200 14463000

MAXINTRINSIC
0003 *01299400*
01299500 09253100 09414120

MAXROW
0003 *01684000*

MAXSAVE
0003 *01598000*

MAXSTACK

0003 *01683000*

13230000 *13231000* 13231000 14277000 *14502000*

MAXSTACK0

0094 *14029000*

14277000 14502000

MAXTLCR

0003 *01331000*

02439000 *02456000*

MCPBIT

0003 *01001000*

01001290 02397000

MCPTOG

0003 *01001290*

MCPTYPE

0003 *01561410*

09430060

MDESC

0069 *09005000*

09034200 09253500 09254000 09255000 09418000 09419100

MEDIUM

0003 *01001580*

02182750 02195000 02195750 *02203250**02204250**02205500**02214100* 02214200 02228750 02421000 05038000
07025020

MERGEBIT

0003 *01001010*

01001300 02425000 *02425500* 02425500 02426000 02427000 02428000 02435000 02436000

MERGEPTION

0016 *02363000*

02428000 *02437000*

MERGETOG

0003 *01001300*

02425500 02427000 02428000 02436000

MIN

0005 *01822000*

01828000 01834000

MKABS

0003 *02022000*

02023000 02202000 02202050 02214000 02215750 02216500 02217750 02226000 02410500 02411000 02456000
02485500 02486000 02903000 05091000 07025010 07025030 08999175 08999200 09400000 09401000

MKS

0003 *01662000*

07408000 07434000 13761000 13765000

MKST

0093 *13750000*

13752000 *13761000* 13761000 13764000

MODE

0003 *01402000*

04175500 04184000 04212100 04223100 05115000 05345000 07397200 07600410 07748000 08093000 14303000
14303000 14478000 *14501000* 14501000

MON

0003 *01147000*

16254000

MONITORDEC

0094 *14014000*

14019000 *14142000*

MONITORV

0003 *01291000*

MOVE

0003 *02054000*

02212250	02366100	02412000	02487000	02602600	02916000	04147000	05039500	07029000	07669000	08999075
08999550	08999575	08999700	09393130	09393170	09393190	09393350	09393400	09393430	09433000	09447060
13281000	13310000	13651000	13746000	14259000	14440000	16495100				

MOVE

0067 *08999075*

08999100 08999275 08999350

MOVECHARACTERS

0003 *02045000*

02705000 07668500 16349000 16392000

MOVECODE

0003 *13683000*

MOVEIT

0021 *02638000*

02704000

MRCLEAN

0003 *01379000*

04164000 *09040000*

MUL

0003 *01663000*

08049000

MULOP

0003 *01278500*

12022000

MYCLASS

0003 *01001590*

02968000 02971000 02974000 *02974500* 02975000 02976500 02977500 *02978500* 02982000 02983000

M0

0040 *05325080*

05325240 05325320

M1

0040 *05325080*

05325250 05325350

M2

0040 *05325080*

05325260 05325360

M3

0040 *05325080*

05325270 05325380

M4

0040 *05325080*

05325280 05325390

M5

0040 *05325080*

05325290 05325400

M6

0040 *05325080*

05325300 05325410

M7

0040 *05325080*

05325310 05325420

N

0003 *01693000*

01759000 02249000 04204000 04237000 05334200 09393000 10258100 13221000 13673150

N

0003 *01737350*
01737350 01737450

N
0003 *01759000*
01781000

N
0003 *02041000*
02041000 02043000

N
0003 *02045000*
02046000 02050000

N
0003 *05325010*
05325460 05325470 05325480

N
0012 *02264700*
02264700 02264850 03036000 03037000 03047000 03047100 03050000 04115000 04116000

N
0012 *02264400*
02264400 02264550

N
0011 *02249000*
02251000 02252000 02255000 02259000

N
0029 *04204000*
04208000 04209000 04221000 04223000

N
0035 *05067000*
05067000 05075000 05246000 05247000 05271000 05298000 05299000

N

0030 *04237000*
04238000 04239000 *04240000* 04240000

N

0040 *05325040*
05325050 05325230

N

0046 *06064000*
06064000 06067000 07035000 07036000

N

0043 *05415000*
05415000 05421000 05437000

N

0041 *05334200*
05334400 05334600

N

0054 *07038100*
07038100 07038500

N

0067 *08999075*
08999075 08999100 08999450

N

0063 *07653500*
07653500 07655500

N

0077 *09393000*
09400000 09402100 09403000 09405500 09406000 09407000 *09413000* 09417000 09420000 *09430050* 09430075

N

0076 *09378000*

09378000 09381000 09393060 09393240

N

0077 *09393700*

09393700 09393730 10228000

N

0077 *09393240*

09393300

N

0087 *13221000*

13258000 13265000 13271000

N

0082 *10257100*

10257100 10257300

N

0082 *10258100*

10263110 10264700 10264900

N

0092 *13673150*

13673300 13673400

NAMEDEC

0094 *14015000*

14020000 *14161000*

NAMEID

0003 *01205200*

07046000 07048000 14161010 14161120 15076100 15276020 15276090 15295100 15303000 15344000 15351000

NAMEV

0003 *01295000*

NCII

0003 *01624000*

13228000 *13228000* 14056000 *14057000*

NCIIO

0094 *14031000*

14056000

NCR

0002 *00511000*

00511000 00512000

NCR

0003 *01330000*

02128500 02129000 02129500 *02213000* *02214000* *02225250* *02229000* 02229000 02230750 *02230750* 02410000
02485000 02665000 02680000 02697000 *02720000* 02731000 02745000 02750000 02753000 02786000 02791000
02807000 02813000 02888000 02902000 *02903000*

NCR

0003 *01756000*

01756000 01761000 01768000 01773000 01780000

NCR

0007 *02089500*

02126000

NCR

0003 *02183500*

02183750 02184500

NCRV

0007 *02090000*

02090500 02092500 02102000 02124500

NEQ

0003 *01664000*

NER

0076 *09364000*

09364000 09368000

NESTBIT

0003 *01001020*

01001310 02472000

NESTCTR

0003 *01722000*

05375100 05375100 *14481200* 14483400

NESTCUR

0003 *01722000*

05402200 05402200 05409100 05409300 *05409300*

NESTFORM

0043 *05415000*

05426000 05431000

NESTLEVEL

0003 *01484000*

16126000 16126000 16142000 *16142000* 16166000 16171000 16260000 16265000

NESTOG

0003 *01001310*

01723000

NESTOG

0003 *01723000*

07397100 14483100 14601000

NESTPRT

0003 *01724000*

05402000 05402200 05402300 05404000 05409200 05409500 05425000 05425400 05430500 05447000 05448000
14483300

NESTS

0103 *16115000*

16144000 16484000

NE\$TSORT

0003 *05411100*

05425200 *05437000* 05442000 05443000

NEW

0003 *02016000*

02018000

NEWBASE

0003 *01001490*

02192000 *02192500**02583000**09028920*

NEWBIT

0003 *01001030*

01001320 02193750 02388000 09282500

NEWINCL

0003 *01001330*

NEWINCLBIT

0003 *01001040*

01001330

NEWINX

0003 *01000860*

NEWTAPBUF

0003 *01490510*

NEWTAPE

0003 *01560000*

02194000 09282500

NEWTGG

0003 *01001320*
02193750 09282500

NEXT
0003 *02956000*
02968000 02968500 02974000 02974500 02978500 15076000

NEXT
0102 *15076000*
15253000 15292000

NEXTINFO
0003 *01620000*
02265350 *02726000* *09033000* 10250000 10254000 10261000 10263000 *10263100* 10285000 *10285000* *13291000*
13302000 13303000 13304000 *13305000* 13308000 13309000 13310000 13311000 13312000 *13312000* 13593000
13594000 13594000 13770600 14055000 14166500 14217000 14319000 14329000 *14330000* *14373000* 14375000
14463000 16442000

NEXTLINK
0062 *07596000*
07604050 07604080 07604090 07607000 07609000

NMI
0003 *01689000*
02681000 *02785000* 02800000 02801000 02833000 02835000 02838000 02840000 12015000

NINFO
0094 *14030000*
14055000 14600000 14601000

NLAB
0101 *14418000*
14422000 *14437000* 14437000

NLO
0003 *01689000*
02681000 *02785000* 02800000 02833000 02835000 02838000 02840000 12014000

NLOC

0101 *14418000*

14422000 14433000 *14433000* 14451200

NLUCS

0094 *14022000*

NODIMPART

0003 *01532000*

07098000 07412000

NOMEADING

0003 *01001480*

01836000 02183000 02195000 02195750 02214200 02228750 02264950 02421000 05039600 05046000 05325490
07025020 *09028050* 09362000 13653500 13676000

NONLITNO

0003 *01209000*

02851000 02924000 07604120 07665000 07666500 08032000 13226000

NONSAVNDX

0077 *09393000*

09405500 09406000 *09407000* 09407000 09427100

NOP

0003 *01665000*

04212100 04223100 07078000 08094000 13752000 16012000

NOP

0103 *16012000*

NORMAL

0032 *04311010*

04311060 04311080 *04311120*

NOTOP

0003 *01272000*

02964000 02974500 06146000

NUMBEREND

0021 *02640000*

02687000 *02845000*

NUMBERS

0007 *02091500*

02096000 02102500 02117000

NUMLE

0065 *08080000*

08095000 08162000 08165000

NXTELB

0003 *01319000*

02644000 02910000 *02912000* 02912000 *02917000**08110000**08174000**09038000* 10258200 10263400 10263500
10263500 10277000 10282000 12112000 *12113000* 12113000 12125000

N1

0005 *01822000*

01828500 01835500

N2

0005 *01822000*

01828500 01835500

OCR

0087 *13221000*

13249000 13250000 13252500 *13257000* 13262000 13263000

OCTALWORDS

0012 *02264400*

02264650 02265100 02265250 02265700 02265950

OCTIZE

0003 *02001838*

02001852 02001854 02686000

OK

0055 *07393100*

07397400 *07397600*

OLDL

0062 *07596500*

07597000 07600800 *07604040* 07604067 07604070 07604100

OLUSEQ

0003 *01741200*

01741200 01741300

OMIT

0003 *02183500*

02183750 02186750

OMITBIT

0003 *01001050*

01001340 02194250 02195000 02195750 02214200 02228750 02235250 02421000 02421100 07025020

OMITTING

0003 *01001340*

02182750 02194250 02195000 02195750 02214200 02228750 02235250 02421000 07025020

OPARSIZE

0003 *01000902*

01000904 02316000 02331000 09414022 09414026

OPCLASS

0025 *02981000*

02982000 02983000 02983500 02984000

OPCLASS

0045 *06042000*

06048000 06052000 06055000

OPCODE

0112 *16366000*

16379000

OPDC

0065 *08016000*

08108000 08130000 08217000

OPERATOR

0003 *01271000*

06042000 06076000

OPERATOR

0003 *04011000*

04014000

OPERATOR

0003 *04006000*

04007000

OPERATOR

0003 *04030000*

04034000

OPERATOR

0045 *06042000*

06044000 06054000

OPERATORS

0003 *01299200*

06071000

OPINX

0003 *01000800*

02311000 02312000 *02312000* 02314000 02316000 02317000 02334000 02338000 02342000 02356000

OPTIONLENGTH

0016 *02364000*

02373000

OPTIONS

0003 *01000904*

02312000 02314000 02317000 02331000 02334000 02338000 02352000 02356000 02426000 09251208 09414024
09414026

OPTIONWORD

0003 *01000910*

01001210 01001220 01001230 01001240 01001250 01001260 01001270 01001280 01001290 01001300 01001310
01001320 01001330 01001340 01001350 01001360 01001370 01001380 01001390 01001400 01001420 01001430
01001440 01001450 01001460 02191500 02193750 02194250 02195000 02195750 02214200 02228100 02228750
02231500 02232500 02235250 *02330000* 02335000 *02335000* *02339000* 02339000 02353000 *02353000* 02357000
02357000 02421000 02425500 02427000 02428000 02436000 02533000 02577000 02602500 04148000 04150000
04278500 04297000 05039600 05046000 05048000 05097500 05099000 05325490 05350100 05376100 06123000
07025020 07086500 09028000 09253050 09253100 09282500 09384500 09387000 09389500 09394100 09405000
09414010 09421000 09430050 09447010 13651100 13652000 13653500 13674000 13676000 13677100 14461500
14493500

OROP

0003 *01275000*

02965000 02984000

OUTDEC

0094 *14014000*

14019000 14158000

OUTPUTSOURCE

0003 *02190500*

02196500 02230500 02235000

OUTV

0003 *01289000*

OWNERR

0094 *14013000*

14018000 *14135000*

OWNV

0003 *01280000*

13383000 13723000 14161040

0003 *01489000*

05414100 12102000 13750000 *14335000* 14387000 14430000 14451200 14463000

P

0003 *02635000*

02644000 02917000 *02917000* 02920000 02923000 02924000

P

0035 *05050000*

05050000 05052000 05055000

P

0035 *05059000*

05059000 05061000

P

0043 *05414100*

05425300 05425400

P

0085 *12102000*

12108000 12126000

P

0093 *13750000*

13752000 13755000 13761000 *13761000* 13761200 *13761200* 13767000

PACK

0003 *04270000*

04276000 04296000

PACKIN

0082 *10258300*

10264000 10264200 10264400 10264420 *10264910*

PACKINFO

0083 *10236000*

10240000 10251000 10254000

PAN

0076 *09364000*

09386000

PANA

0003 *03021000*

06186000 *06413000* 06417000

PARM

0098 *14254200*

14254700 14259080

PARSE

0003 *03014000*

06035000 06181000 *06313000* 06335000

PASSFORMAT

0003 *05214000*

05227000

PCTR

0054 *07038000*

07040000 07042000 *07094000* 07094000 07098000

PDES

0003 *01299020*

13746000

PEN

0076 *09370000*

09377000 09388000

PERCENT

0021 *02637000*

02641000 *02731000* 02906000

PERIOD

0003 *01264000*

06105000 06106100 06340000 07021000 07026000 15111000 15233003 15276140 15339000

PERMANENT

0003 *05333000*

05345000

PINF00

0094 *14026000*

14375000 14471000 14483000 14489000

PJ

0003 *01617000*

13344000 *13344000* 14073000 14085000 14117000 *14120000* 14120000 *14323000* 14333300 14334000 14335000
14349000 14364000 14374000 *14387000* 14411300 14415300

PLUG

0065 *08020000*

08116000 08188000 08213000 08218000

PN

0093 *13750600*

13751100 *13761110* 13770000 *13770600* 13773000

POINTER

0087 *13219000*

13222000 13223000 13226000 13229000 13240000 *13240000* 13249000 13255000 13257000 13263000 13279000
13281000 13284000 13286000 *13287000* 13287000 13290000

POL

0054 *07037000*

07052000 07059000

POLISHER

0003 *03009000*

06060000 06170000 07052000 07512000 07743000

POLISHV

0003 *01212100*

07052000 07511000 13338000

PORS

0003 *05325010*

05325460 05325470 05325480

PORS

0040 *05325040*

05325050 05325090 05331000 05333000

POSITION

0003 *04270000*

04271000 04273000 04311000 04500000 04503000 04505000 05008000

PRIMARY

0003 *03005000*

06016000 06018000 06023000 06035000 06049000 *06126000* 06147000 06172000 06181000 06182000 07076000

PRINT

0092 *13661000*

13673000 13675000

PRINT

0091 *13633000*

13643000 13653000

PRINTCARD

0003 *02182500*

02195000 02195750 02214200 02228750 02421000 07025020

PRINTDOLLARBIT

0003 *01001060*

01001350 02228750 02377000

PRINTDOLLARTOG

0003 *01001350*

02228750

PRL
0003 *01666000*

PROAD
0094 *14032000*
*14313000**14320000* 14461100 14476000 14489000

PROADD
0003 *01613000*
14037000 *14476000* 14602000

PROCEDUREDEC
0094 *14015000*
14020000 14137700 *14270000*

PRUCID
0003 *01187000*
07399000 07404000 13278000 14284000 14292000

PROCSTMT
0003 *03029000*
06162000 *07391000* 07425000 07735000

PRUCV
0003 *01293000*

PRUGDESCBLDR
0003 *03047000*
05245000 05255000 07677000 13252500 13746000 14461100

PROGRAM
0003 *09003000*
09448000 16495200

PRGINFO
0003 *01608000*
07090000 13677200 13677300 14304000 *14314000**14320000* 14388100 14411200 14411300 14415200 14415300

14457000 14461200 14470000 14483200 14483300 14483400 *14496000* 15080000

PRT

0003 *01343000*

05248000 05252000 09414040 09415000

PRTAD

0003 *13734000*

13746000

PRTADR

0003 *05247000*

05247500 05247500 05248000 05252000

PRTBASE

0003 *01342000*

01343000 01724000 01725000 05424000 09039000 09414030 09415000 09419100

PRTBIT

0003 *01001070*

01001360 02395000 05376100

PRTE

0003 *01667000*

04018000 04023000 04033000 05316000 06110300 08068000 15106100 15310000

PRTI

0003 *01703000*

05349000 07600440 *09039000* 14022000 14054000

PRTIMAX

0003 *01703000*

05347000 *05350000* 05350000 *09039000* 09388000 09414100 09419000

PRTIO

0094 *14030000*

14054000

PRTOG

0003 *01001360*

05376100

PRTOP

0003 *01342000*

01343000 01724000 01725000 05424000

PRTSIZ

0076 *09370000*

09371000 09372000

PTUG

0003 *01592000*

13341100 13343000 13370000 *14332000**14373000**14388000*

PUNCHBIT

0003 *01001080*

01001370 02506000 05048000

PUNCHTOG

0003 *01001370*

05048000

PURGE

0003 *03068000*

13215000 13745000 13773000 14471000 14483000 14601000

PURGEBIT

0003 *01001090*

01001380 02508000

PURGETOG

0003 *01001380*

PURPT

0003 *01628000*

02726000 13285000 13309000 14088000 14118000 14446000

PUSHEE

0077 *09393240*

09393480 09396000 09402100 09406000

PUSHER

0077 *09393050*

*09393140**09393200* 09393230 09424000 09427500

PUT

0003 *05008000*

05132000	06104400	07545000	07610000	09133900	09134100	13304000	13348000	13365000	13408000	13768000
14116000	14117000	14168000	14224000	14259150	14315000	14334000	14353000	14388100	14411200	14415200
14430000	14451200	14457000	14470000	15376400	16138000	16166000	16167000	16172000	16257000	16260000

PUTNBUMP

0003 *03057000*

09034000	*13589000*	14167000	14195000	14217000	14221000	14319000	14331000	14403000	16000600	16443000
16445000	16446000									

PUTOGETHER

0082 *10234000*

10257000 10263900 10268000 10269000 10270000 10273000 10274000

PUTSEQNO

0003 *02005000*

02006000 02202060 02214250 02218000 02234750 02368000 02716000 02904000

P1

0003 *03057000*

P1

0003 *15070000*

15102000	15104000	15110100	15114000	15119000	15233008	15233015	15276100	15276110	15276160	15276190
15276210	15305000	15334000	15335000	*15336000*	15342000	15349000	15364000	15367000	*15370000*	

P1

0035 *05050000*

05096000

P2

0003 *01588000*

05059000 13341000 13343000 13347000 *13385000* 13722000 *13723000**14133000**14161070**14318000**14321000*

P2

0035 *05059000*

05096000

P3

0003 *01589000*

13725000 *13726000**14133000**14138000**14139000**14140000**14141000*

Q

0003 *01316000*

01822000 02280000 02313000 02314000 *02327000* 02343000 *02371000* 02375000 02376000 02378000 02382000
02384000 02386000 02389000 02392000 02394000 02396000 02400000 02406000 02420000 02422000 *02424000*
02433000 02462000 02467000 02469000 02471000 02473000 02477000 02479000 02481000 02495000 02497000
02499000 02505000 02507000 02509000 02515000 02519000 02521000 02523000 02529000 02566000 02576000
02588000 *02590000**02650000**02686500**02696000**02698000* 02700000 *02754000**02822000**02846000**02865000*
02879000 *02958500* 02960500 02961000 02964000 02964500 02965000 02965500 04311020 05043000 05044000
05343000 05414100 07086200 07086300 08015000 09393010 *13282000**14441000* 16213000 16378500

Q

0002 *00523000*

00523000 00524000 00526000 01716000 01717000 01717700 01717900 01718000 01719000

Q

0005 *01822000*

01833000 *01833000**01834000*

Q

0003 *04231000*

04239000 04241000 04243000

Q

0032 *04311020*

*04311050**04311070* 04311070 04311090

Q

0043 *05414100*
05425100 05425100 *05425200* 05425200 05425300

Q

0041 *05343000*
*05344750**05371000**05374000* 05376100

Q

0065 *08015000*
08093000 08094000 08185000 08200000 08210000 08218000

Q

0077 *09393010*
09395700 09405500 *09405500* 09447020

Q

0082 *10257100*
10257100 10257700

QUOTE

0021 *02637000*
02641000 *02690000*

R

0003 *02183500*
02183750 02185750 02202500

RANGE

0003 *05297000*
05304000

RCA

0103 *16008000*

RDV

0003 *01668000*

READACARD

0003 *02065000*

02130500 *02196750* 02238000 02731000 09035000

READTAPE

0009 *02198500*

02202250 02208500 02219500

REALARRAYID

0003 *01203000*

07056000 07057000 13380000 13389000

REALDEC

0094 *14013000*

14018000 14135000 14136000 *14138000*

REALID

0003 *01199000*

07060000 07069000 08038000 12035000 14138000 15276210

REALPROCID

0003 *01195000*

REALSTRPROCID

0003 *01191000*

14294000

REALSUBID

0003 *01184000*

14164000 14167000

REALV

0003 *01283000*

13389000 14161120 14164000

RECOV

0087 *13220000*

13251000 13256000 *13281000*

RED

0003 *02011000*

02011000

REED

0003 *08999000*

08999175 08999425 09399000

REGD

0065 *08010000*

08015000 *08093000* 08094000 *08111000* 08120000 *08160000* *08170000* 08185000 08200000 *08210000* 08218000
08231000

RELAD

0003 *13734000*

RELAD

0094 *14029000*

14487000 14489000

RELOP

0003 *01277000*

05344500 14260000 16207000

REMCOUNT

0082 *10230000*

10245000 10247000 *10248000* *10249000* 10249000 *10263000*

REPEAT

0003 *04011000*

04013000 04014000

RESIZE

0003 *02056000*

RESTORESEQNUM

0016 *02320200*

02602800

RESULT

0003 *01386000*

02128500 02129000 02278000 *02327000* 02366200 *02371000* *02499000* 02500000 02579000 02586000 *02590000*
 02591000 02598000 *02602700* *02647000* 02648000 *02663000* *02666000* *02667000* *02684000* *02694000* *02698000*
 02700000 *02744000* *02747000* *02749000* *02751000* *02752000* *02754000* *02757000* *02788000* *02793000* *02806000*
 02809000 *02812000* *02816000* *02820000* *02846000* *02889000* *02890000* *02958500* 02959000 02959500 10265000
 10272000 10276000

RESULT

0007 *02089500*

02092500

RESULTSWITCH

0021 *02643000*

02648000

RESULTV

0007 *02089500*

02090500 02094000

RETURNSTORE

0065 *08010000*

08087000 08090000 *08166000* *08168000* *08190000* 08193000 *08207000* 08215000

RIGHT

0003 *03039500*

07675500 *08999450* 08999800 14461300 14603000

RINX

0088 *13301000*

13302000 13304000 13305000

ROSE

0021 *02639000*

02875000 02879000 02881000

ROUND

0062 *07594000*

07600000 07600900 07604000 *07645000*

RQW

0041 *05344000*

RP

0047 *06137000*

RRB1

0003 *01522000*

RRB2

0003 *01525000*

RR1

0002 *00507000*

00529000 *00530000* 01557000 *14329000* 14373000

RR10

0002 *00507000*

RR11

0002 *00507000*

00529000 00531000 00533000 00536000

RR2

0002 *00507000*

00530000 01557000

RR3

0002 *00507000*
*00531000**00532000* 01558000

RR4
0002 *00507000*
00532000 01558000

RR5
0002 *00507000*
*00534000**00535000* 01560000

RR6
0002 *00507000*
*00534000**00535000* 01560000

RR7
0002 *00507000*
*00534000**00535000* 01560000

RR8
0002 *00507000*
*00537000**00538000* 01561000

RR9
0002 *00507000*
*00537000**00538000* 01561000 07595000 *07599000* 07601000

RSA
0103 *16018000*

RTBRKET
0003 *01265000*
05278000 06141000 06322000 06422000 10264300 12126000 13395000 13402000 14259120 15260000 15276010
15294000

RTN
0003 *01669000*
13740000

RTPAREN

0003 *01266000*

02639000 02976500 06115000 06175000 06416000 07067000 07096000 10264300 12039000 12126000 13756000
13761100 13763000 14259120 14336000 16132000 16478000

RTPAREN

0021 *02639000*

02642000 *02744000*

RTS

0003 *01670000*

07087000 13264000 13268000

S

0003 *02001858*

02001868

S

0003 *02001838*

02001842

S

0003 *03051001*

S

0003 *03067000*

S

0003 *04291000*

04296000 04297000

S

0003 *04277500*

04280000 04280500 04281500 04282000 04283500 04284000 04285000 04285500

S

0012 *02264400*

02264500

S

0012 *02264700*

02264800 02307000

S

0033 *04502000*

04548000 04551000

S

0039 *05309000*

05309000 05311000 05325010

S

0042 *05401000*

05407000 *05407000*

S

0047 *06132000*

06138000

S

0043 *05416000*

05418000 05420000

S

0041 *05334100*

05334300 05334500

S

0046 *06063000*

06074000 *06074000**06103000* 06103000

S

0065 *08027000*

08029000

S
0065 *08063000*
08065000 08069000

S
0064 *07719000*
07727000

S
0065 *08072000*
08075000 08079000

S
0084 *12004000*
12006000 12025000 *12025000* 12030000 *12030000**12038000* 12038000 12055000 *12055000*

S
0082 *10257100*
10257200 10257800

S
0085 *12103000*
12104000 13216000 13217000 13315000 13316000 13590000 13591000 13632000

S
0085 *12102000*
12110000

S
0098 *14254200*
14254300 15070000 16029000 16065000 16085000

S
0092 *13673100*
13673200 13673350

S

0103 *16029000*

16032000

SAF

0003 *01300200*

13677400 *14299000*

SAN

0003 *01677100*

SAV

0003 *05247000*

05248000

SAVECODE

0003 *13632000*

13651000 13651100 13653000

SAVECODE

0091 *13633000*

13633000 13638000 13657000

SAVEINFO

0090 *13378000*

13394000 13396000 *13397000**13401000* 13405000 *13405000* 13408000

SAVEINX

0017 *02324000*

02331000 *02342000* 02352000

SAVEL

0003 *01623000*

13606000 *13628000* 14055000 *14609000*

SAVELO

0094 *14030000*

14055000 14609000

SAVERR

0094 *14013000*

14018000 *14136000*

SAVETIME

0002 *00503000*

SAVEV

0003 *01281000*

13726000 14292000 14298000 14299000

SAVINFO

0077 *09392300*

09395100 09396000 09402100 09414070 09414110 09418000 09419000 09419100 09424000 09433000

SAVL

0104 *16031000*

16032000 16034000

SAVL

0106 *16087000*

16088000 16096000

SAVNDX

0077 *09393000*

09397000 09402100 *09403000* 09403000 09413000 *09414120* *09420000* 09426000 09436000

SBIT

0003 *07036000*

07046000

SBV

0003 *01244000*

16197000 16415000

SCAN

0007 *02089500*

02127000 02128500

SCAN

0082 *10257100*

10264800

SCANAGAIN

0021 *02639000*

02646000 02650000 02686500 02696000 02728000 02737000 02890000

SCANNER

0003 *02066000*

02137000 02279000 02327000 02371000 02499000 02590000 02647000 02663000 02666000 02684000 02694000
02698000 02744000 02747000 02749000 02751000 02752000 02754000 02788000 02793000 02806000 02809000
02812000 02816000 02820000 02889000 02890000 02958500

SCATTERELBAT

0003 *03035000*

07395000 *13197000* 13208000 13327000 13767000 14206000 14305000

SCI

0003 *01677050*

SCLASS

0054 *07038000*

07044000 *07046000* 07048000 07056000 07060000 *07079000* 07088000

SCOUNT

0016 *02365100*

02366200 02602700

SCRAM

0003 *01301000*

02865000 *09385000* 09386000 12110000 *12122000* 13308000 13361000 13362000 13755000

SCS

0003 *01677150*

SCV

0003 *01246000*

16206000 16337000

SEARCH

0003 *04231000*

04243000 04244000 04280000 04280500 04283500 04284000

SEC

0103 *16022000*

16342000

SECOND

0003 *05271000*

05281000 05282000

SECOND

0051 *06314000*

06325000 06328000 06332000

SECOND

0052 *06339000*

06341000 06344000

SECRET

0003 *01628000*

13331000 14333000 14345000

SED

0103 *16017000*

SED

0101 *14419000*

SEGMENT

0003 *03042000*

07676500 *13657000* 13681000 14461200 14602000

SEGMENTSTART

0003 *03041000*

07661500 *13632000* 13655000 14298000 14387000

SEGSBIT

0003 *01001100*

01001390 02470000 13652000 13674000

SEGSIZEMAX

0003 *01687000*

13678000 13678000

SEGSTOG

0003 *01001390*

13652000 13674000

SEMICOLON

0003 *01230000*

05109000 07012000 07032000 10264410 13766000 14062000 14063000 14165000 14325000 14338000 14357000
14519000 16131000

SEQ

0003 *01756000*

SEQ

0003 *04130000*

04130000 04132000 04142000 04203000 04231000 04247000

SEQ

0054 *07038100*

07038300 07038600

SEQ

0076 *09378000*

09382000 09393050 09393240

SEQBIT

0003 *01001110*

01001400 02191500 02393000

SEQCOMPARE

0009 *02202500*

02209750 02224500

SEQERRBIT

0003 *01001120*

01001420 02520000

SEQERRTOG

0003 *01001420*

SEQTOG

0003 *01001400*

02191500

SEQUENCEERROR

0003 *01742100*

01742110 02226750

SEQXEQTOG

0003 *01001560*

02525000 02525000

SES

0101 *14419000*

SET

0085 *12103000*

12117000

SETTING

0003 *01000802*
02327000 02338000 02352000 02356000 02427000

SGAVL
0003 *01369000*

SGND
0003 *01370000*
14054000

SGND0
0094 *14030000*
14054000

SIGNA
0065 *08012000*
08014000 *08056000* 08065000 08098000 08108000 08130000 08179000 08181000 08189000 08212000 08217000
08227000

SIGNB
0065 *08012000*
08050000 08112000 *08121000* 08126000 08183000 08196000 *08203000* 08214000

SIGNC
0065 *08012000*
08014000 08044000 *08112000* 08185000 08197000 *08202000* 08219000 *08220000*

SIMPARITH
0003 *03003000*
06006000 *06032000* 07085000

SIMPG0
0003 *03026000*
07548000 *07555000* 07557000 07564000 07570000 07581000

SIMPI
0065 *08052000*
08061000 08061000 08176000 08211000 08224000

SIMPLE

0065 *08027000*

08033000 *08038000* 08039000 08112000 08181000 08183000 08185000

SIMPLEB

0065 *08014000*

08044000 08112000 08220000

SIMPLEV

0065 *08015000*

08074000 08099000 08211000 08224000 08227000

SINGLBIT

0003 *01001130*

01001430	02195000	02195750	02214200	02228750	02421000	02522000	04150000	04278500	05039600	05046000
05097500	05099000	05325490	06123000	07025020	07086500	09384500	09387000	09389500	13653500	13676000

SINGLT0G

0003 *01001430*

02181000	02195000	02195750	02214200	02228750	02421000	04150000	04278500	05039600	05046000	05097500
05099000	05325490	06123000	07025020	07086500	09384500	09387000	09389500	13653500	13676000	

SIV

0003 *01236000*

16319000 16320000 16328000 16334000

SIZE

0003 *03042000*

SIZE

0003 *05247000*

05248500 05249000 05250000 05252000 05254000

SIZE

0003 *07647000*

07663000 07675500 07676500 07677000

SIZE

0003 *13657000*

13675000 13677400 13678000

SIZE

0092 *13661000*

13661000 13670000

SKAN

0003 *02277000*

02327000 02371000 02499000 02590000 02958500

SKANAGAIN

0016 *02360000*

02370000	02378000	02383000	02385000	02389000	02465000	02468000	02478000	02503000	02527000	02577250
02584000	02595000	02599000								

SKBIT

0063 *07653500*

07653500 07655000 08020000 08027000 08083000 08084000

SKIPCOUNT

0083 *10241000*

10247000 10253000 10254000 10256000

SKIPIT

0024 *02974000*

02974500 02978500

SKIPS

0103 *16403000*

16419000 16492000

SKIPV

0003 *01241000*

SKP

Q003 *02001838*

02001838 02001842

SKP

0003 *02041000*

02041000 02043000

SKP

0003 *02001858*

02001858 02001890

SKSC

0082 *10259000*

10263500 10283000

SN

0093 *13750000*

13755500 13767000 13768000 *13769000* 13769000

SND

0003 *01671000*

15104000 15233015 15276110 15335000

SOP

0003 *14003000*

14036000 14506000 14600000

SORCE

0003 *02045000*

02048000

SORTNEST

0003 *05412000*

14601000

SORTPRT

0003 *01725000*

05425100 05425400 05447000 05448000 05452000 05455000 05459000

SPACEITDOWN

0094 *14023100*

14461500 14493500

SPCLMON

0102 *15076200*

15349000 15351300 15351400 15354000

SPECIAL

0003 *01003000*

02655000 02666000 02755000 02961500 02963000 07031000 09198000

SPECIALCHAR

0021 *02637000*

02643000 *02651000*

SPECIALSWITCH

0021 *02641000*

02659000

SPECTOG

0003 *01593000*

05034000 13305100 13328000 13385000 13601000 13621000 13719000 14070000 *14122000* 14142000 14149000
14160000 14161070 14187000 14205000 14275000 14287000 *14364000**14368000* 14377000 14507000

SPRT

0003 *01698000*

SRESULT

0016 *02365100*

02366200 02602700

SSKIP

0003 *02045000*

02046000 02049000

SSP

0003 *01672000*
05226000

SSS
0046 *06063500*
06068500 06125000

STACKCT
0003 *01566010*
04527000 *06054150* 06158000 *06158000*06172000* 06172000 *06174500* 06174500 *06179500* 06179500 *06296500*
*06299500**06305000**06332500**07041200**07100500**07726990**14125500**15091000**15098000**15101500**15233011*
15233011 *15233016**15234500**15276115**15278000**15335500**15369000*

STACKCTR
0003 *01683000*
05370000 *05371000* 05371000 06068500 *06125000* 14058000 14477000 *14478000* 14481200 *14493000* 14506000
14610000

STACKCTRD
0094 *14029000*
*14058000**14477000* 14493000 14610000

STACKHEAD
0003 *01310000*
02724000 02865000 05090000 05233000 05234000 09134100 09134200 09405500 09447030 09447040 13286000
13308000 13361000 13362000 13768000 13769000 14448000 16443000

START
0066 *08084000*
08086000

START
0094 *14017000*
14063000 14138000 14139000 14140000 14141000 14156000 14160000 14162000 14170000 14199000 14202000
14253000 14254050 14269000 14272000 14379000 14384000 14504000 14521000

START
0097 *14202000*

START
0099 *14272000*
14373000

START
0098 *14254050*
14259090 *14269000*

START
0100 *14384000*
14503000

START
0103 *16475000*
16477000 16485000

STARTINTRSC
0069 *09024000*

START1
0099 *14272000*
14325000 14364000 *14375000*

START2
0099 *14273000*
14376000

STD
0003 *01673000*
08090000 12038000 13230000 15104000 15233015 15276110 15335000

STLPI
0003 *05003000*
05003000 05108000 05274000 05275000 05276000 05277000 05278000 05344500 05344520 05344610 05344670
06069000 06071000 06085000 06094000 06106100 06199000 06317000 06318000 06319000 06320000 06321000
06322000 06415000 06421000 07018000 07041000 07437000 07505000 07552000 07567000 07597500 07600100
07600600 07604110 07604140 07645000 07662000 07664000 07666500 07673000 07727070 07767000 07768100
07994000 08029000 08178000 09252000 12007000 12010000 12013000 13351000 13392000 13395000 13402000
13404000 13753000 13754000 13756000 13766000 13771000 14126000 14220000 14223000 14259110 14268000

14338000	14341000	14345000	14356000	14393000	14396000	14467000	15090000	15113000	15233000	15233006
15253000	15260000	15276050	15276160	15300000	15342000	16125000	16160000	16197000	16206000	16207000
16208000	16217000	16220000	16252000	16253000	16316000	16319000	16324000	16326000	16340000	16367000
16368000	16372000	16384000	16406000	16434000	16435000					

STEPIT

0003 *05002000*

05344690	06015000	06035000	06049000	06102000	06104700	06106200	06111000	06117000	06140000	06143000
06147000	06166000	06168000	06172000	06174000	06177000	06181000	06201000	06208000	06303000	06331000
06345000	06410000	06411000	06416000	06417000	06422000	06423000	07010000	07072000	07100000	07407000
07485000	07487000	07493000	07495000	07505000	07516500	07518000	07552000	07569000	07573000	07580000
07584000	07604100	07727020	07727090	07996000	08031000	08039000	08120000	08131000	08135000	08142000
08155000	08161000	08175000	08192000	08231000	12013000	12017000	12025000	12030000	12034000	12036000
12039000	12045000	12054000	13327000	13406000	13409000	13753100	13757000	13758000	13761100	13772000
14166000	14193000	14259000	14259070	14259140	14314000	14359000	14403000	14472000	15098000	15233012
15274000	15276070	15306000	15376800	16130000	16176000	16234000	16376000	16408000	16412000	16494000

STEPV

0003 *01255000*

08103000 08182000

STLABID

0003 *01178000*

14403000 14447000 16000300 16253000 16437000 16443000 16481000

STLB

0003 *01471000*

15273000

STMT

0003 *03027000*

05108000	07011000	07485000	07495000	07574000	07577000	07585100	*07711000*	07771000	08163000	08201000
09275000	14167000	14488000	14515000							

STMTSTART

0065 *08010000*

08157000 *08169000* *08191000* 08193000 *08206000* 08221000

STUP

0100 *14384000*

14484000 *14491000*

STUPDEFINE

0003 *01597000*

02893000 *02911000**02926000**07017000**10263300**10281000**12111000**13327000**14130000**14192000**14258000*
*14259060**14259100**14259130**14402000**14403000*

STOPENTRY

0003 *01594000*

13351000 *14188000**14198000**14205000**14251000**14255000**14269000**14318000**14321000**14402000**14404000*

STUPGSP

0003 *01694000*

13343000 13345000 *14164500**14166000**14188000**14198000**14205000**14251000**14255000**14269000**14318500*
*14319500**14402000**14404000*

STOPPER

0003 *03068000*

STOPPER

0003 *13217000*

13223000 13291000

STOPPER

0003 *13734000*

13745000

STORE

0065 *08063000*

08070000 08130000 08135000 08138000 08216000

STORE

0066 *08084000*

08089000

STOREFIX

0066 *08091000*

08093000 08157000 08171000

STREAMERR

0094 *14017000*

14021000 *14137000*

STREAMSTMT

0003 *03040000*

07011000 13771000 14415400 *16001000* 16130000 16229000 16235000 16495000

STREAMTOG

0003 *01417000*

02692000 02692500 04279500 04281000 04283000 04284500 05344400 07011000 *07661000* *07676000* 12110100
12110100 *12127100* 12127100 13343000 *13771000* *13773000* 14077000 14082000 *14280000* 14298000 14318500
14324000 14361000 14377000 14385000 *14461000*

STREAMV

0003 *01297000*

07745000 07768100 14278000

STREAMWORDS

0003 *05230000*

05236000 13771000 13773000 14324000 14460000

STRMPROCSTMT

0003 *03030000*

06160000 *07426000* 07446000 07737000

STRNGCON

0003 *01211000*

02705100 07075000 07604110 07665500 07668000 08032000 16210000 16384000

STRNGXT

0021 *02638000*

02701000

STRPROCID

0003 *01189000*

07431000 14281000

STUFF

0003 *01561600*

05350300 09281000 13677500

STUFFBIT

0003 *01001140*

01001440 02516000 05350100 13677100

STUFFF

0003 *05199000*

05203000

STUFFTOG

0003 *01001440*

05350100 13677100

ST1

0007 *02090000*

02090500 02120000 02124500 02126000

ST2

0007 *02090000*

02090500 02121000 02121500

SUB

0003 *01674000*

06016000 08030000 08044000 08214000 12015000 16329000

SUBC

0003 *13213000*

SUBDEC

0094 *14014000*

14019000 *14163000*

SUBHAND

0003 *03011000*

06156000 *06194000* 06209000 07733000

SUBID

0003 *01185000*

06198000 14164000

SUBLEVEL

0003 *01402000*

05117000 *14477000* 14477000 *14492000*

SUBOP

0003 *01556500*

05344570 05344710

SUBV

0003 *01288000*

SUPERFRMTID

0003 *01183000*

05221000 05224000

SVARMONFILE

0003 *01529000*

SW

0008 *02191250*

02194500

SW

0017 *02326000*

02328000

SWITCHDEC

0094 *14015000*

14020000 *14200000* 14252000

SWITCHID

0003 *01186000*

07507000 14207000

SWITCHIT

0016 *02321000*

02359000	02377000	02388000	02393000	02395000	02397000	02402000	02415000	02421100	02425000	02435000
02470000	02472000	02474000	02480000	02490000	02496000	02506000	02508000	02511000	02516000	02520000
02522000	02531000	02567000	02600000							

SWITCHV

0003 *01292000*

SWITMONFILE

0003 *01539000*

SYLLABLE

0003 *03028000*

SYLLABLE

0003 *03019000*

SYLLABLE

0003 *04270000*

04271000 04274000 04311000 04500000

SYMBOL

0003 *02183500*

02183750 02185250

T

0003 *01717900*

T

0003 *02057000*

02058000 02059000

T
 0003 *01564000*
 02057000 02249000 02323000 *02655000* 02658000 *02666000* *02692000* 02695000 *02701000* 02705100 *02721000*
 02722000 *02724000* 02726000 *02755000* 02798000 *02813000* 02823000 *02823000* 02824000 02826000 *02826000*
 02827000 02842000 *02848000* 02850000 02851000 *02865000* 02875000 02876000 *02877000* 02877010 *02877020*
 02878000 *02882000* 02882000 02886000 02896000 *02896000* 02896000 02900000 02901000 02910000 02957500
 02981500 05068000 05096000 *05233000* 05234000 05236000 05401000 05414000 *08057000* 08057000 08058000
 08059000 08061000 *08069000* 08070000 *09282600* *09388000* 09388000 09389000 09393020 12004000 13199000
 14163500 16067000 16364000

T
 0003 *03050000*

T
 0003 *04311000*
 04311030 04312000 04318000

T
 0003 *12101000*
 12108000 12117000

T
 0018 *02446000*
 02446000 02448000 02449000 03024000 03025000 03036000 03037000 03042000 03047000 03047100 03050000
 03051000 03051001 04010000 04011000 04029000 04030000 04115000 04116000

T
 0018 *02441000*
 02441000 02443000 02444000

T
 0017 *02323000*

T
 0011 *02249000*
 02251000 *02257000* *02259000* 02259000 02260000

T
 0025 *02981500*

02982500 02983000 02983500 02984000 02984500

T

0022 *02957500*

02960500 *02961000* 02961000 02961500 *02964000* 02966500

T

0035 *05068000*

05074000 05081000 05086000

T

0043 *05414000*

05425400 05428000 05428000 *05430500* 05431000

T

0042 *05401000*

05402000 05402100 05403000 05404000 05405000 05409000 *05409300* 05409300 *05409400* 05409400 05409500

T

0067 *08999075*

08999075 08999100

T

0076 *09364000*

09364000 09369000

T

0077 *09393020*

T

0084 *12004000*

12044000 12047000

T

0086 *13199000*

13200000 13201000 13202000 13203000 13204000 13205000 13206000 13207000

T
0096 *14163500*
14166500 14168000

T
0105 *16067000*
16070000

T
0112 *16364000*
16393000 16394000

TABLE
0003 *02635000*
02920000 *02924000* 02927000 05002000 05003000 05221000 05224000 07066000 07397210 07401000 07600950
07604020 07748000 07751000 07768130 10263400 10282000 12020000 12112000 13753100 13761100 14063000
14516000 15276010 16482300

TAKE
0003 *01717700*
02724000 02725000 *03053000* *05005000* *05006000* 05124000 05131000 05189000 06082000 06104300 06191000
07042000 07090000 07098000 07412000 07441000 07527000 07529000 07539000 07601000 09133900 09134100
10263110 13226000 13229000 13249000 13255000 13257000 13279000 13284000 13286000 13348000 13362000
13364000 13365000 13408000 13677200 13767000 14084000 14088000 14118000 14168000 14259150 14310000
14320000 14333300 14352000 14388100 14411200 14415200 14425000 14430000 14438000 14446000 14448000
14451200 14470000 14483200 14483400 15376300 16070000 16089000 16138000 16139000 16163000 16166000
16172000 16254000 16260000 16449000

TAKEFRST
0003 *05188000*
05189000 05222000 06197000 07435000

TALL
0102 *15072000*
15077000 15080000 *15083000* 15083000 15085000 15086000 15092000 15094000 15097000 15102000 15105000
15106100 15107000 15110200 15110300 15124000 15128100 15233009 15233010 15233011 15233014 15233018
15255000 15267000 15276020 15276060 15276080 15276090 15276210 15290000 15295100 15302000 15303000
15310000 15333000 15344000 15344100 15344200 15351000 15351200 15352000 15353000 15376300

TALLYV
0003 *01239000*
16319000 16341000

TAN

0103 *16014000*

16213000

TAPE

0003 *01561000*

02201750 02452000 02454000 02457000 09282000

TAPE

0003 *02188000*

02188000 02188500 02202750

TAPELAST

0009 *02210000*

02210750 *02219250*

TBUFF

0003 *01561056*

02201750 02202000 02456000 02457000 07025030

TB1

0003 *01457000*

*07506000**07507000* 07516200 07518000 *10243000**10260000* 10275000 14204000

TB1

0097 *14204000*

14206000

TCLASS

0111 *16314000*

16315000 16317000 16324000 16328000 16334000 16341000 *16348000* 16349000 16350000 16352000

TCOUNT

0003 *01566000*

02251000 02252000 *02758000**02790000* 02796000 02798000 02803000 *02810000**02817000*

TEDOC

0003 *13683000*

13686000 13703000 13706000 13732000 13734000

TEMPL

0028 *04166000*

04172000 04175000 04184500 04186000

TEMP1

0003 *02001860*

02001866 02001868 02001890

TEMP2

0003 *02001860*

TEN

0003 *01340000*

02796000 02799000 02803000 02826000 02833000 02835000 02838000 02840000 09041000

TEST

0065 *08041000*

08051000 08131000 08220000

TESTCODE

0109 *16192000*

16211000 16212500

TESTLEV

0100 *14384100*

14387000 14389000 14411300 14415300 14451000 14461100

TESTVOID

0009 *02210500*

02232500 *02234000*

THENBRANCH

0050 *06295000*

06297000 06301000

THENV

0003 *01256000*

06411000 16217000

THERE

0003 *01737350*

01737450

THI

0003 *01689000*

02250000 02255000 02256000 02257000 02785000 02797000 02799000 02800000

THIRD

0051 *06314000*

06326000 06327000 06328000 06332000

TIME1

0003 *01300000*

01828000 09385000 *16495200*

TL

0027 *04118000*

04119000 04124000

TL

0057 *07483000*

07485000 07487000

TL

0087 *13221000*

13258000 13277000

TLCR

0003 *01330000*

02202050 02202060 02202070 02224500 *02456000*

TLCR

0009 *02202750*

02203500 02204000

TLU

0003 *01689000*

02250000 02255000 02256000 02785000 02796000 02797000 02799000 02800000

TOGGLEV

0003 *01245000*

TOLDC

0069 *09005000*

09006000

TOP

0044 *05439000*

05445000 *05452000*

TOTALNO

0003 *01000860*

02192000 *02192500* 02193000 *02193000**02582000*

TOV

0003 *01257000*

07505000 07552000 16252000 16437000

TOWARDS

0003 *04116000*

04119500 04120100 04120200 04120500

TOWARDS

0003 *03037000*

TRB

0003 *01682000*

04318000

TRANSFER

0003 *01251000*

16379000

TRP

0103 *16019000*

16383000

TRUTHV

0003 *01208000*

TRW

0101 *14419000*

TSSINTYPE

0003 *01561430*

09430060

TSSTOG

0077 *09393020*

09414026 09430050

TSUBLEVEL

0094 *14029000*

14477000 14492000

TURNONSTOPLIGHT

0003 *02011000*

02202070 02214500 02218250 02235750 02369000 02717000 02905000

TWXA

0003 *01561700*

05350180 05350300 13677400 13677500

TYPE

0003 *03055000*

TYPE

0003 *05247000*

05248000 05249000 05250000 05253000

TYPE

0003 *13316000*

13333500 13338000 13342000

TYPE

0003 *13716000*

13729000

TYPE

0050 *06295000*

06299000

TYPE

0103 *16476000*

16477000

TYPEV

0003 *01611000*

*13380000**13389000* 13390000 *14091000**14161010**14161120* 14162000 14163500 *14281000**14284000* 14285000
14292000 14294000 *14294000* 14319000

TYPEV

0003 *03071000*

TYPEV

0096 *14163500*

14164000 14165000 14167000

T1

0003 *01693000*

02001860 04278000 04502000 05049000 06062000 06184000 06196000 07427000 07562000 08173000 *13381000*

13383000 13389000 13684000 *14161000**14161020* 14161040 14161120 15073000

T1
0003 *02001860*

02001874 02001876

T1
0003 *13684000*

13685000 13707000 13711000 13732000 13734000

T1
0033 *04502000*

04550000 04551000

T1
0031 *04278000*

04280500 04281000 04281500 *04284000* 04284500 04285000

T1
0035 *05049000*

05088000 *05091000* 05091000 05096000 05097000 *05097000**05099000* 05099000

T1
0046 *06062000*

06072000 06077000 *06079000* 06080000 06081000 06082000 06087000 06089000 *06104200* 06104600 *06107200*
06108000 06110000 06110100

T1
0048 *06184000*

06185000 06187000 *06189000* 06191000

T1
0049 *06196000*

06197000 06206000 06207000

T1
0056 *07427000*

07435000 07436000 *07440000* 07442000 07443000

T1
0065 *08173000*
08179000 *08193000* 08202000 08203000 08204000 08205000 08206000 08207000 08225000

T1
0061 *07562000*
07566000 07572000 07573000 07575000 *07577000* 07579000 07583000 07585100

T1
0102 *15073000*
15095000 15097000 15098000 15100000 15112000 15124000 15233005 15233013 15233018 15276060 15276070
15276150 15276210 15304000 15308000 15341000 15370000 *15376300* *15376500* 15376500 15376600 *15376600*
15376700

T2
0003 *01693000*
02001860 04502000 05049000 06062000 06184000 07562000 07657500 08173000 13684000 15074000

T2
0003 *02001860*
02001874 02001876

T2
0003 *13684000*
13686000 13694000 13698000

T2
0035 *05049000*
05092000 05093000 05095000 05098000

T2
0033 *04502000*
04550000

T2
0048 *06184000*
06185000 06189000 *06190000* 06190000 06191000

T2

0046 *06062000*

06072000 06074000 06075000 06079000 *06080000* 06080000 06081000 *06084000* 06087000 06089000 06090000
06090000 06092000 *06093000* 06099100 06099200 *06104300* 06104400 *06106000* *06106200* 06110200

T2

0065 *08173000*

08198000 08208000

T2

0063 *07657500*

07663000 07668500 07669000 07671500

T2

0061 *07562000*

07583000 07583000 *07585000* 07586000

T2

0102 *15074000*

15100000 15112000 15124000 15233005 15233013 15233018 15276070 15276150 15276210 15308000 15341000
15370000 *15376300* 15376400

T3

0003 *13684000*

13689000 13697000 13700000 13710000 13732000 13734000

T3

0035 *05049000*

05091000 05092000 *05093000* 05094000 05096000 *05098000*

T3

0065 *08173000*

08199000 08209000

T4

0035 *05049000*

05095000 05096000

T4

0065 *08173000*

08200000 08210000

U
0003 *05411100*

U
0003 *05437000*
05440000 05441000 05443000 05446000 05454000 05457000 05458000

UNHOOK
0003 *01626000*
13305300 *13353000*

UNKNOWNID
0003 *01177000*
06151000

UNTILV
0003 *01226000*
07018000 07485000 08113000 08128000 08184000

UPPER
0003 *05299000*
05304000

USEROPINX
0003 *01001170*
02335000 02339000 02353000 02357000 02600000 09414022

USESWITCH
0009 *02210750*
02212000

USETHESWITCH
0009 *02210250*
02211750 02228100 02228150 02229500 02231500 02232250 02234000 02234600 02235250

V

0065 *08010000*

08015000 08078000 08099000 08181000 08188000 *08225000**08227000*

VAL

0003 *01741200*

01741300

VALUEV

0003 *01258000*

14341000

VARIABLE

0003 *03038000*

06140000 06164000 07050000 07058000 07073000 07083000 07402000 07739000 08228000 12037000 *15070000*
15377000

VBIT

0054 *07039000*

07043000 07045000

VD

0003 *01150000*

07043000 13203000

VOIDBIT

0003 *01001150*

01001450 02231500 02415000

VOIDCR

0003 *01785000*

02231750 02232000 *02232000**02410500* 02410500 02411000

VOIDING

0003 *01001450*

02231500

VOIDPLACE

0003 *01785000*

02232000 02410500 02412000

VOIDRANGE

0016 *02365200*

02410000 02411000 02412000 02485000 02486000 02487000

VOIDTAPE

0003 *01001460*

02228100 02232500

VOIDTBIT

0003 *01001160*

01001460 02228100 02232500 02490000

VOIDTCR

0003 *01785000*

02228125 02228150 02233000 02233500 *02233500* 02485500 *02485500* 02486000

VOIDTPLACE

0003 *01785000*

02233500 02485500 02487000

VONF

0003 *01590000*

*13203000**13341000* 13373000 *16000400*

VP

0003 *01756000*

01761000 01780000

VRET

0065 *08010000*

08054000 08077000 *08176000* 08198000 *08208000* 08211000 08224000 *08228000*

W

0003 *02054000*

02054000 02055000

W

0003 *01717700*

W

0039 *05309000*

05311000 05315000 05325010

W

0092 *13673100*

13673100 13673250

WD

0069 *09005000*

09005000 09006000

WHAT

0078 *09393080*

09393170 09393180 09393190

WHATISIT

0016 *02362000*

02364000 02365000 02379000 02398000 02475000 02517000 02568000 *02578000*

WHILESTMT

0003 *07490000*

07496000 07757000

WHILEV

0003 *01224000*

08113000 08133000 08139000

WHU

0078 *09393080*

09393110 09393130 09393170 *09393190* 09393200

WHOLE

0056 *07427000*

07430000 07432000 07438000 07439000 *07441000* 07441000 07442000 07444000

WITHV
0003 *01259000*

WOP
0003 *01371000*
02534000 02536000 04280500 04281000 04281500 04284000 04284500 04285000

WORD
0003 *04142000*
04147000 04149000

WORD
0003 *04270000*
04273000 04277500 04291000 04311000 04500000 04503000 04504000 05005000 05008000 05012000

WORD
0003 *05008000*
05009000

WORDCOUNT
0088 *13301000*
13302000 13307000 13310000 13312000

WRITEAX
0054 *07038100*
07086300

WRITELINE
0003 *02181000*
02183000 02195000 02195750 02214200 02228750 02421000 04150000 05039600 05046000 05097500 05099000
05325490 06123000 07025020 07086500 09384500 09387000 09389500 13653500 13676000

WRITEOUT
0046 *06064000*
06120000

WRITEPRT

0003 *05325010*

05325500 05376100

WRITERROR

0034 *05016000*

05031000 05044000

WRITNEW

0003 *02016000*

02020000 02194000

X

0003 *13591000*

13593000

X

0035 *05067000*

05072000 05074000 05105000 05113000 05123000 05128000 05199000 05207000 05246000 05247000 05271000
05298000 05299000 05307000

X

0079 *09393270*

09393400 09393410 09393420 09393430

XBIT

0016 *02321000*

02327000 02335000 02339000 02353000 02357000

XCH

0003 *01675000*

04507000 06203000 08065000 08123000 08228000 12054000 15106100 15309000

XIT

0003 *01676000*

02210500 05325080 13743000

XIT

0009 *02210500*

02213250 02214750 *02237750*

XIT

0040 *05325080*

05325340 05325350 05325370 05325380 05325390 05325400 05325410 05325430

XITR

0003 *01430000*

XMAS

0077 *09393050*

09393110 09393130 09393140 09393170 09393180 *09393180* 09393190 09393200

XMODE

0003 *01001530*

02328000 *02332000**02367000**02383000**02385000* 02408000 02425500 *02478000* 02483000 02525000 02577000

XMODE0

0017 *02325000*

02326000 *02329000*

XMODE1

0017 *02325000*

02326000 *02333000*

XMODE2

0017 *02325000*

02326000 *02337000*

XMODE3

0017 *02325000*

02326000 *02341000*

XMODE4

0017 *02325000*

02326000 *02355000*

Y

0077 *09393240*

09393290 *09393330* 09393350 09393400 *09393430* 09393440 09393450

Z

0003 *01490000*

06063500 13746000 *14411200**14415200* 14457000 14506000 *14506000*

Z

0046 *06063500*

06099100

ZEERO

0063 *07658500*

07660500 07662000

ZP1

0003 *01677000*

LABEL OLPA OLPA 00176366CC USER=00000000; REMOVE ESPOL/DISK;COMPILE ESPOL/DISK TSPOL LIBRARY;TSPD TSPOL /ESPUL